Intel® Pentium® III Xeon™ Processor Specification Update

Release Date: April 2005

Order Number: 244460-040

The Pentium® III Xeon™ processor may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

CONTENTS

REVISION HISTORY

Date of Revision	Version	Description
March 1999	-001	This document is the first Specification Update for the Pentium® III Xeon™ processor.
April 1999	-002	Removed Erratum G15. Renumbered remaining errata. Added Errata G45 and G46. Corrected Errata table “Plans” column for G29 and G42. Updated the Pentium III Xeon Processor Identification and Package Information table. Moved revised Mixed Steppings statement to the General Information section.
June 1999	-003	Added Errata G47 and G48. . Added Documentation Change G1. Added Specification Clarifications G1 and G2. Added Specification Change G2. Updated the Pentium III Xeon Processor Identification and Package Information table.
July 1999	-004	Added Errata G49 and G50. Added footnote 7 on Pentium III Xeon Processor Identification and Package Information table. Changed the “Plans” column in the Summary Table of Changes from “Fix” to “Fixed” for G35, G36, G37, G44, and G46. Updated Documentation Changes, Specification Clarifications, and Specification Changes introduction paragraphs.
August 1999	-005	Added Errata G51 and G52. Added Documentation Change G2. Updated Codes Used in Summary Table. Updated column heading in Errata, Documentation Changes, Specification Clarifications and Specification Changes tables.
September 1999	-006	Added new Boxed Processors to the Pentium III Xeon Processor Identification and Package table. Revised Erratum G52.
October 1999	-007	Added Brand ID column to *Identification Information*. Added Errata G53.
November 1999	-008	Revised the *Pentium® III Xeon™ Processor Identification Information* table. Added errata G54, G55, and G56. Added *Documentation Change* G3. Updated *Identification Information, Pentium® III Xeon™ Processor Identification Information, and Mixed Steppings.* Added Ca2 column to *Summary of Errata, Summary of Documentation Changes, Summary of Specification Clarifications, and Summary of Specification Changes.*
December 1999	-009	Added Errata G57 - G65. Added Documentation Change G4. Added Specification Change G3. Updated documentation reference in Preface. Updated Summary of Errata table for errata G19 and G33. Updated L2 cache size table in *Identification Information* section.
January 2000	-010	Added 800/133 MHz Pentium® III Xeon™ processor (CPUID 68*x*h) information in the *Pentium® III Xeon™ Processor Identification Information* section. Changed references to the processor stepping to the format “CPUID/Stepping”. Revised Erratum G25. Added Errata G66, G67, and G68. Added Documentation Change G5.
February 2000	-011	Updated *Pentium® III Xeon™ Processor Identification Information* table. Revised Erratum G55 and G68. Added Specification Changes G4 and G5. Added Documentation Change G6. Updated Summary of Changes product letter codes.
March 2000	-012	Updated *Identification Information, Pentium® III Xeon™ Processor Identification Information and Mixed Steppings* Information. Updated Preface reference information. Updated the CPUID/Stepping information in the Summary of Changes section. Revised errata G43, G52, G54, G57, G58, G59, G60, G61, G62, G66 and G68. Added erratum G69.
April 2000	-013	Changed the “Plans” column in the *Summary of Errata* to “Fixed” for G57. Changed affected steppings for errata G34, G57, G65, and G66.
May 2000	-014	Updated references in Preface. Updated the Identification Information tables. Updated entries in the *Pentium® III Xeon™*

REVISION HISTORY

Date of Revision	Version	Description
		Processor Identification Information tables. Updated *Summary of Errata, Summary of Documentation Changes, Summary of Specification Clarifications and Summary of Specification Changes* tables. Updated Erratum G19 and G50. Added Errata G71 - G74.
May 2000	-015	Special Launch Edition. Updated the *Pentium® III Xeon™ Processor Identification and Package Information* table.
June 2000	-016	Added Erratum G75. Added Specification Change G6.
July 2000	-017	Changed the "Plans" column in the *Summary of Errata* from "NoFix" to "Fix" for G49 and G63. Changed the "Plans" column in the *Summary of Errata* from "Fixed" to "Fix" for G43, G52, G54, G57, G58, G60, G61, and G62 (to clarify that while fixed on processors with certain CPUIDs, will be fixed on others). Changed the "Plans" column in the *Summary of Errata* from "Fix" to "NoFix" for G33 (to clarify that erratum only affects processors with CPUID 67xh). Changed the "Plans" column in the *Summary of Errata* from "Fixed" to "NoFix" for G64 (to clarify that erratum only affects processors with CPUID 67xh). Revised Errata G40 and G75. Added Errata G76 and G77.
August 2000	-018	Updated entries in the *Pentium® III Xeon™ Processor Identification Information* tables. Added 6Axh/A1 column to *Summary of Errata, Summary of Documentation Changes, Summary of Specification Clarifications, and Summary of Specification Changes.* Changed the "Plans" column in the *Summary of Errata* from "Fix" to "Fixed" for G57, G70, G71, G72, and G74. Added Erratum G78. Added Documentation Change G7. Added Specification Clarification G3.
August 2000	-019	Special Launch Edition. Updated entries in the *Pentium® III Xeon™ Processor Identification Information* tables. Added 68xh/C0 column to *Summary of Errata, Summary of Documentation Changes, Summary of Specification Clarifications, and Summary of Specification Changes.* Changed the "Plans" column in the *Summary of Errata* to "Fixed" for Errata G19, G33, G34, G50, and G64. Changed affected steppings for Erratum G19 (documentation error). Revised Errata G28, G54, and G67. Added Errata G79 and G80. Added Documentation Changes G8 and G9.
September 2000	-020	Updated entries in the *Pentium® III Xeon™ Processor Identification Information* tables.
October 2000	-021	Updated the document list in the Preface. Added Erratum G81. Added Documentation Changes G10 and G11.
November 2000	-022	Added Erratum G82.
December 2000	-023	Corrected core stepping and CPUID information in *Pentium® III Xeon™ Processor Identification Information* table for S-Specs SL4PZ and SL4R9. Updated Specification Update product key to include the Intel® Pentium® 4 processor, Revised Erratum G2. Added Documentation Changes G12 through G17.
January 2001	-024	Revised Erratum G2. Added Documentation Changes G18 and G19.
February 2001	-025	Revised Documentation Change G18. Added Documentation Change G20.
March 2001	-026	Updated the Pentium® III Xeon™ Processor Identification Information and MP Platform Population Matrix for the Pentium® III Xeon™ Processor at 100 MHz System Bus tables, updated Summary of Errata, Summary of Documentation Changes, Summary of Specification Clarification and Summary of Specification Changes tables, added errata G83 and G84.
March 2001	-027	Added erratum G85.
May 2001	-028	Updated the Pentium® III Xeon™ Processor Identification Information MP Platform Population Matrix for the Pentium® III Xeon™ Processor at 100 MHz System Bus tables. Changed the "Plans" column in the *Summary of Errata* table from "Fix" to "Fixed" for Erratum G85.

REVISION HISTORY

Date of Revision	Version	Description
July 2001	-029	Added Boxed Processors S-Spec SL4U2 to *Pentium III Xeon Processor Identification and Package* table. Corrected the *MP Platform Population Matrix for the Pentium® III Xeon™ Processor at 100 MHz System Bus* table in Part I.
August 2001	-030	Added erratum G86.
November 2001	-031	Added Documentation Changes G1-G5.
January 2002	-032	Added Documentation Changes G6-G10. Added new S-Spec parts for 700MHz.
March 2002	-033	Deleted old info from Documentation Change table.
April 2002	-034	Added Erratum G87. Edited Erratum G25. Added Document Changes G1-G3. Minor changes throughout document.
May 2002	-035	Updated Erratum G25. Revised Doc Change Table and added G1-G3. Updated *Summary of Errata* table and descriptions
June 2002	-036	Added doc changes G1-G11 Added Erattum G88
July 2002	-037	Renamed G4AP to G25 in the errata table. Removed G4AP Re-named CPUID to Processor Signature.
November 2003	-038	Added Erratum G4AP and G5AP
March 2005	-039	Added Errata G89-G97. Updated Errata G86 and G5AP.
April 2005	-040	Updated letter code list in Summary Table of Changes. Added Specification Clarification G1.

PREFACE

This document is an update to the specifications contained in the following documents:

- *Pentium® III Xeon™ Processor at 500 and 550 MHz* datasheet (Order Number 245094)
- *Pentium® III Xeon™ Processor at 600 MHz to 1 GHz with 256KB L2 Cache* datasheet (Order Number 245305)
- *Pentium® III Xeon™ Processor at 700 MHz with 1MB and 2MB L2 Cache* datasheet (Order Number 248711)
- *Intel Architecture Software Developer's Manual*, *Volumes 1, 2 and 3* (Order Numbers 245470, 245471, and 245472, respectively)
- *P6 Family of Processors Hardware Developer's Manual* (Order Number 244001)

It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains S-Specs, Errata, Documentation Changes, Specification Clarifications and Specification Changes.

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc., as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Pentium® III Xeon™ processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that processor are present on all devices unless otherwise noted.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

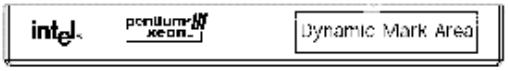
Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

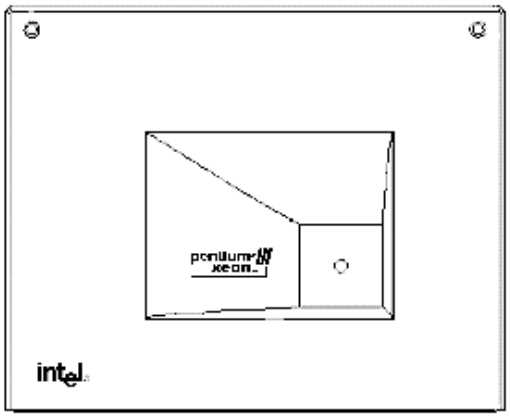
Specification Changes are modifications to the current published specifications for the Pentium III Xeon processor. These changes will be incorporated in the next release of the specifications.

Specification Update for the Pentium® III Xeon™ Processor

GENERAL INFORMATION

Pentium® III Xeon™ Processor and Boxed Pentium® III Xeon™ Processor Markings



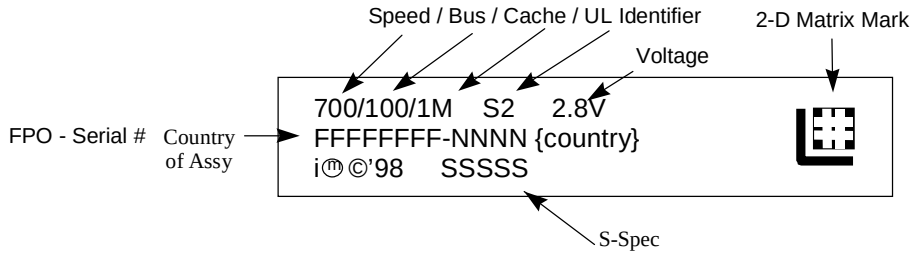


Production Dynamic Mark Example:

700/100/1M S2 2.8V
FFFFFFFF-NNNN {country}
i©'98 SSSSS

2D Matrix Contents Example:
Intel 80526KY7001M
FFFFFFFF-NNNN

Dynamic Mark Area



IDENTIFICATION INFORMATION

The Pentium® III Xeon™ processor can be identified by the following values:

Family[1]	Model[2]	Brand ID[3]
0110	0111	00h = Not Supported
0110	1000	03h = "Intel® Pentium® III Xeon™ Processor"
0110	1010	03h = "Intel® Pentium® III Xeon™ Processor"

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after Reset, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after Reset, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.

The Pentium III Xeon processor's second level (L2) cache size can be determined by the following register contents:

512-Kbyte Unified L2 Cache[1]	43h
1-Mbyte Unified L2 Cache[1]	44h
2-Mbyte Unified L2 Cache[1]	45h
256-Kbyte, 8-way set associative, 32-byte line size, L2 Cache[1]	82h
1-Mbyte 8 way set associative 32byte line size, L2 Cache[1]	84h
2-Mbyte 8 way set associative 32byte line size, L2 Cache[1]	85h

NOTE:

1. For the Pentium® III Xeon™ processor, the L2 cache size corresponds to a token in the EDX register after the CPUID instruction is executed with a 2 in the EAX register. Other Intel microprocessor models or families may move this information to other bit positions or otherwise reformat the result returned by this instruction; generic code should parse the resulting token stream according to the definition of the CPUID instruction.

Mixed Steppings in MP Systems

Intel Corporation fully supports mixed steppings of Pentium III Xeon processors. The following list and processor matrix describes the requirements to support mixed steppings:

- Mixed steppings are only supported with processors that have identical family and model number as indicated by the CPUID instruction.
- While Intel has done nothing to specifically prevent processors operating at differing frequencies from functioning within a multiprocessor system, there may be uncharacterized errata that exist in such configurations. Intel does not support such configurations. In mixed stepping systems, all processors must operate at identical frequencies (i.e., the highest frequency rating commonly supported by all processors).
- While there are no known issues associated with the mixing of processors with differing cache sizes in a multiprocessor system, and Intel has done nothing to specifically prevent such system configurations from operating, Intel does not support such configurations since there may be uncharacterized errata that exist. In mixed stepping systems, all processors must be of the same cache size.
- While Intel believes that certain customers may wish to perform validation of system configurations with mixed frequency or cache sizes, and that those efforts are an acceptable option to our customers, customers would be fully responsible for the validation of such configurations.
- The workarounds identified in this and following specification updates must be properly applied to each processor in the system. Certain errata are specific to the multiprocessor environment and are identified in the *Mixed Stepping Processor Matrix* found at the end of this section. Errata for all processor steppings will affect system performance if not properly worked around. Also see the “Pentium III Xeon Processor Identification and Package Information” section for additional details on which processors are affected by specific errata.
- In mixed stepping systems, the processor with the lowest feature-set, as determined by the CPUID Feature Bytes, must be the Bootstrap Processor (BSP). In the event of a tie in feature-set, the tie should be resolved by selecting the BSP as the processor with the lowest stepping as determined by the CPUID instruction.
- *Functional Redundancy Checking Mode* (FRC mode), a feature of Pentium III Xeon processors with CPUID 67*x*h, is not supported using a master and checker pair of processors with different stepping, model number, cache size, or frequency.

In the following processor matrix, “NI” indicates that there are currently no known issues associated with mixing these steppings. A number indicates that a known issue has been identified as listed in the table following the matrix. A multiprocessor system using mixed processor steppings must assure that errata are addressed appropriately for each processor.

MP Platform Population Matrix for the Pentium® III Xeon™ Processor at 100 MHz System Bus

Processor Signature /Core Stepping	67*x*h/B0	67*x*h/C0	6A*x*h/A0	6A*x*h/A1	6Axh/B0
67*x*h/B0	NI	1, 2	Not Supported	Not Supported	Not Supported
67*x*h/C0	1, 2	NI	Not Supported	Not Supported	Not Supported
6A*x*h/A0	Not Supported	Not Supported	3	1	1, 4
6A*x*h/A1	Not Supported	Not Supported	1	NI	1, 4
6Axh/B0	Not Supported	Not Supported	1, 4	1, 4	4

NOTES:

1. Some of these processors are affected by errata, which may affect the features an MP system is able to support. See the “Pentium III Xeon Processor Identification and Package Information” table for details on which processors are affected by these errata.
2. CPUID 67*x*h/B0 and CPUID 67*x*h/C0 stepping processors may have differing cache components which require differing voltages. When mixing these steppings the proper cache voltage as identified by the VID pins must be supplied to each cartridge.
3. See Note 11 and Note 12 in the Pentium III Xeon Processor Identification and Package Information table.
4. The Pentium III Xeon processor at 900 MHz with 2MB of L2 cache will ignore the logic states presented to the core/bus ratio pins (A20M#, IGNNE#, LINT0, and LINT1) at the de-assertion of the RESET# signal, and will operate only with a 9:1 core/bus ratio. For this reason, the Pentium III Xeon processor at 900 MHz with 2MB of L2 cache should only be used in systems containing identical processors. Use of the Pentium III Xeon processor at 900 MHz with 2MB of L2 cache in systems containing the Pentium III Xeon processor at 700 MHz with 2MB of L2 cache will result in processors running at different frequencies, which is not a supported configuration.

DP Platform Population Matrix for the Pentium® III Xeon™ Processor with 133-MHz System Bus			
Processor Signature/Core Stepping	**68*x*h/A2**	**68*x*h/B0**	**68*x*h/C0**
68*x*h/A2	NI	1	1
68xh/B0	1	NI	1
68xh/C0	1	1	NI

NOTES:

1. Some of these processors are affected by errata, which may affect the features an MP system is able to support. See the “Pentium III Xeon Processor Identification and Package Information” table for details on which processors are affected by these errata.

Pentium® III Xeon™ Processor Identification and Package Information

S-Spec Number	Step	Processor Signature	Speed Core/FSB (MHz)	L2 Size (Kbytes)	Cache and Stepping	Processor Substrate Revision	Cartridge Revision	Notes
SL2XU	B0	0672h	500/100	512	C6C B0	512K-6A	2.0	1, 2, 5, 7
SL2XV	B0	0672h	500/100	1024	C6C B0	1M-6A	2.0	1, 2, 5, 7
SL2XW	B0	0672h	500/100	2048	CK1 B1	2M-Ka	2.0	1, 2, 5, 7
SL3C9	B0	0672h	500/100	512	C6C B0	512K-6A	2.0	1, 2, 3, 5, 7
SL3CA	B0	0672h	500/100	1024	C6C B0	1M-6A	2.0	1, 2, 3, 5, 7
SL3CB	B0	0672h	500/100	2048	CK1 B1	2M-Ka	2.0	1, 2, 3, 5, 7
SL3FK	C0	0673h	550/100	512	CK2 B2	512K-Ka	2.0	2, 4, 7, 8
SL3D9	C0	0673h	500/100	512	C6C B0	512K-6A	2.0	2, 7
SL3DA	C0	0673h	500/100	1024	C6C B0	1M-6A	2.0	2, 7
SL3DB	C0	0673h	500/100	2048	CK1 B1	2M-Ka	2.0	2, 6, 7
SL3AJ	C0	0673h	550/100	512	CK2 B1	512K-6A	2.0	2, 7, 8
SL3CE	C0	0673h	550/100	1024	CK1 B1	1M-6A	2.0	2, 7, 8
SL3CF	C0	0673h	550/100	2048	CK1 B1	2M-Ka	2.0	2, 6, 7
SL3TW	C0	0673h	550/100	1024	CK1 B1	2M-Ka	2.0	2, 7
SL3Y4	C0	0673h	550/100	512	CK2 B2	512K-Ka	2.0	2, 7
SL3FR	C0	0673h	550/100	512	CK2 B2	512K-Ka	2.0	2, 3, 4, 7, 8
SL385	C0	0673h	500/100	512	C6C B0	512K-6A	2,0	2, 3, 7
SL386	C0	0673h	500/100	1024	C6C B0	1M-6A	2,0	2, 3, 7
SL387	C0	0673h	500/100	2048	CK1 B1	2M-Ka	2,0	2, 3, 6, 7
SL3LM	C0	0673h	550/100	512	CK2 B2	512K-Ka	2.0	2, 3, 7, 8
SL3LN	C0	0673h	550/100	1024	CK1 B1	1M-Ka	2.0	2, 3, 7, 8
SL3LP	C0	0673h	550/100	2048	CK1 B1	2M-Ka	2.0	2, 3, 6, 7
SL3BJ	A2	0681h	600/133	256	N/A	cB	2.0	9
SL3BK	A2	0681h	600/133	256	N/A	cB	2.0	10
SL3BL	A2	0681h	667/133	256	N/A	cB	2.0	9
SL3DC	A2	0681h	667/133	256	N/A	cB	2.0	10
SL3SF	A2	0681h	733/133	256	N/A	cB	2.0	9
SL3SG	A2	0681h	733/133	256	N/A	cB	2.0	10
SL3V2	A2	0681h	800/133	256	N/A	cB	2.0	10
SL3V3	A2	0681h	800/133	256	N/A	cB	2.0	10
SL3SS	A2	0681h	600/133	256	N/A	cB	2.0	3, 10
SL3ST	A2	0681h	667/133	256	N/A	cB	2.0	3, 10
SL3SU	A2	0681h	733/133	256	N/A	cB	2.0	3, 10
SL3VU	A2	0681h	800/133	256	N/A	cB	2.0	3, 10
SL3WM	B0	0683h	600/133	256	N/A	cB	2.0	9
SL3WN	B0	0683h	600/133	256	N/A	cB	2.0	10
SL3WP	B0	0683h	667/133	256	N/A	cB	2.0	9
SL3WQ	B0	0683h	667/133	256	N/A	cB	2.0	10
SL3WR	B0	0683h	733/133	256	N/A	cB	2.0	9
SL3WS	B0	0683h	733/133	256	N/A	cB	2.0	10

Pentium® III Xeon™ Processor Identification and Package Information

S-Spec Number	Step	Processor Signature	Speed Core/FSB (MHz)	L2 Size (Kbytes)	Cache and Stepping	Processor Substrate Revision	Cartridge Revision	Notes
SL3WT	B0	0683h	800/133	256	N/A	cB	2.0	9
SL3WU	B0	0683h	800/133	256	N/A	cB	2.0	10
SL3WV	B0	0683h	866/133	256	N/A	cB	2.0	9
SL3WW	B0	0683h	866/133	256	N/A	cB	2.0	10
SL3WX	B0	0683h	933/133	256	N/A	cB	2.0	9
SL3WY	B0	0683h	933/133	256	N/A	cB	2.0	10
SL4H6	C0	0686h	733/133	256	N/A	cB	2.0	9, 16
SL4H7	C0	0686h	733/133	256	N/A	cB	2.0	10, 16
SL4H8	C0	0686h	800/133	256	N/A	cB	2.0	9, 16
SL4H9	C0	0686h	800/133	256	N/A	cB	2.0	10, 16
SL4HA	C0	0686h	866/133	256	N/A	cB	2.0	9, 16
SL4HB	C0	0686h	866/133	256	N/A	cB	2.0	10, 16
SL4U2	C0	0686h	866/133	256	N/A	cB	2.0	3, 10, 16
SL4PZ	B0	0683h	866/133	256	N/A	cB	2.0	3, 10, 16
SL4HC	C0	0686h	933/133	256	N/A	cB	2.0	9, 16
SL4HD	C0	0686h	933/133	256	N/A	cB	2.0	10, 16
SL4R9	C0	0686h	933/133	256	N/A	cB	2.0	3, 10, 16
SL4HE	C0	0686h	1 GHz/133	256	N/A	cB	2.0	9, 17
SL4HF	C0	0686h	1 GHz/133	256	N/A	cB	2.0	10, 17
SL3U4	A0	06A0h	700/100	1024	N/A	Xa	2.0	11, 13
SL3U5	A0	06A0h	700/100	1024	N/A	Xa	2.0	11, 14
SL3WZ	A0	06A0h	700/100	2048	N/A	Xa	2.0	11, 13
SL3X2	A0	06A0h	700/100	2048	N/A	Xa	2.0	11, 14
SL4GD	A0	06A0h	700/100	1024	N/A	Xa	2.0	12, 13
SL4GE	A0	06A0h	700/100	1024	N/A	Xa	2.0	12, 14
SL4GF	A0	06A0h	700/100	2048	N/A	Xa	2.0	12, 13
SL4GG	A0	06A0h	700/100	2048	N/A	Xa	2.0	12, 14
SL49P	A1	6A1h	700/100	1024	N/A	Xa	2.0	13, 15
SL49Q	A1	6A1h	700/100	1024	N/A	Xa	2.0	14, 15
SL49R	A1	6A1h	700/100	2048	N/A	Xa	2.0	13, 15
SL49S	A1	6A1h	700/100	2048	N/A	Xa	2.0	14, 15
SL4RZ	A1	6A1h	700/100	1024	N/A	Xa	2.0	3, 13, 15
SL4R3	A1	6A1h	700/100	2048	N/A	Xa	2.0	3, 13, 15
SL4XU	B0	6A4h	700/100	1024	N/A	xA	2.0	13, 15
SL5D4	B0	6A4h	700/100	1024	N/A	xA	2.0	3, 13, 15
SL4XV	B0	6A4h	700/100	1024	N/A	xA	2.0	14, 15
SL4XW	B0	6A4h	700/100	2048	N/A	xA	2.0	13, 15
SL5D5	B0	6A4h	700/100	2048	N/A	xA	2.0	3, 13, 15
SL4XX	B0	6A4h	700/100	2048	N/A	xA	2.0	14, 15
SL4XY	B0	6A4h	900/100	2048	N/A	Xa	2.0	13, 15, 18
SL4XZ	B0	6A4h	900/100	2048	N/A	Xa	2.0	14, 15, 18
SL5D3	B0	6A4h	900/100	2048	N/A	Xa	2.0	3, 13, 15, 18

NOTES:

1. These processors are affected by Erratum G44.
2. The performance-monitoring event counter 1 may be inaccurate when counting events for the Data Cache Unit and External Bus Logic in these processors. Use counter 0 to count these events.
3. This is a boxed processor with attached passive heatsink.
4. These processors are validated for use in two-way systems only.
5. These processors are affected by Erratum G46.
6. These processors are affected by Erratum G48.
7. Performance-monitoring event counters do not reflect MOVD and MOVQ stores to memory on these processors.
8. These processors are affected by Erratum G56.
9. These processors are designed to operate at 2.8V and have been tested to 55°C T_{PLATE}.
10. These processors are designed to operate at either 5V or 12V and have been tested to 55°C T_{PLATE}.
11. These processors implement an AGTL+ reference voltage of 11/15 * V_{TT} (1.1V when V_{TT} = 1.5V nominal), as explained in Erratum G71. Intel does not support mixing these processors with those designated with Note 12.
12. These processors implement an AGTL+ reference voltage of 11.5/15 * V_{TT} (1.15V when V_{TT} = 1.5V nominal), and an R_{TT} value of 130Ω. Intel does not support mixing these processors with those designated with Note 11.
13. These processors are designed to operate at 2.8V, and have been tested to 65°C T_{PLATE}.
14. These processors are designed to operate at either 5V or 12V, and have been tested to 65°C T_{PLATE}.
15. These processors implement an AGTL+ reference voltage of 11/15 * V_{TT} (1.1V when V_{TT} = 1.5V nominal).
16. These processors operate with a core voltage of 1.68V.
17. These processors operate with a core voltage of 1.70V.
18. These processors ignore the logic states presented to the core/bus ratio pins (A20M#, IGNNE#, LINT0, and LINT1) at the de-assertion of the RESET# signal, and will operate only with a 9:1 core/bus ratio.

SUMMARY OF CHANGES

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Pentium III Xeon processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Erratum, Documentation Change, Specification Clarification, or Specification Change applies to the given processor stepping.
(No mark) or (blank box):	This item is fixed in or does not apply to the given stepping.
PlanFix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Doc:	Intel intends to update the appropriate documentation in a future revision.
PKG:	This column refers to errata on the Pentium III Xeon processor substrate.
AP:	APIC related erratum.
Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor

B = Mobile Intel® Pentium® II processor

C = Intel® Celeron® processor

D = Intel® Pentium® II Xeon™ processor

E = Intel® Pentium® III processor

F = Intel® Pentium® 4 processor Extreme Edition

G = Intel® Pentium® III Xeon™ processor

H = Mobile Intel® Celeron® processor at 466/433/400/366/333/300 and 266 MHz

K = Mobile Intel® Pentium® III Processor

L = Intel® Celeron® D processor

M = Mobile Intel® Celeron® processor

N = Intel® Pentium® 4 processor

O = Intel® Xeon™ processor MP

P = Intel® Xeon™ processor

Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90 nm process technology

R = Intel® Pentium® 4 processor on 90 nm process

S = 64-bit Intel® Xeon™ processor with 800 MHz system bus (1 MB and 2 MB L2 cache versions)

T = Mobile Intel® Pentium® 4 processor-M

V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package

W = Intel® Celeron-M processor

X = Intel® Pentium® M processor on 90 nm process with 2 MB L2 cache

Y = Intel® Pentium® M processor

Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B 0	C 0	A 2	B 0	C 0	A 0	A 1	B 0			
G1	X	X	X	X	X	X	X	X		NoFix	FP data operand pointer may be incorrectly calculated after FP access which wraps 64-Kbyte boundary in 16-bit code
G2	X	X	X	X	X	X	X	X		NoFix	Differences exist in debug exception reporting
G3	X	X	X	X	X	X	X	X		NoFix	FLUSH# servicing delayed while waiting for STARTUP_IPI in MP systems
G4	X	X	X	X	X	X	X	X		NoFix	Code fetch matching disabled debug register may cause debug exception
G5	X	X	X	X	X	X	X	X		NoFix	Double ECC error on read may result in BINIT#
G6	X	X	X	X	X	X	X	X		NoFix	FP inexact-result exception flag may not be set
G7	X	X	X	X	X	X	X	X		NoFix	BTM for SMI will contain incorrect FROM EIP
G8	X	X	X	X	X	X	X	X		NoFix	I/O restart in SMM may fail after simultaneous MCE
G9	X	X	X	X	X	X	X	X		NoFix	Branch traps do not function if BTMs are also enabled
G10	X	X								NoFix	Checker BIST failure in FRC mode not signaled
G11	X	X								NoFix	BINIT# assertion causes FRCERR assertion in FRC mode
G12	X	X	X	X	X	X	X	X		NoFix	Machine check exception handler may not always execute successfully
G13	X	X	X	X	X	X	X	X		NoFix	LBER may be corrupted after some events
G14	X	X	X	X	X	X	X	X		NoFix	BTMs may be corrupted during simultaneous L1 cache line replacement
G15	X	X	X	X	X	X	X	X		NoFix	Near CALL to ESP creates unexpected EIP address
G16	X	X	X	X	X	X	X	X		NoFix	Mixed cacheability of lock variables problematic in MP systems
G17	X	X	X	X	X	X	X	X		NoFix	MCE due to L2 parity error gives L1 MCACOD.LL
G18	X	X	X	X	X	X	X	X		NoFix	Memory Type field undefined for nonmemory operations
G19	X	X								Fixed	Infinite snoop stall during L2 initialization of MP systems
G20	X	X	X	X	X	X	X	X		NoFix	FP data operand pointer may not be zero after power on or reset
G21	X	X	X	X	X	X	X	X		NoFix	Premature execution of a load operation prior to exception handler invocation
G22	X	X	X	X	X	X	X	X		NoFix	EFLAGS discrepancy on page fault after multiprocessor TLB shootdown

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B 0	C 0	A 2	B 0	C 0	A 0	A 1	B 0			
G23	X	X	X	X	X	X	X	X		NoFix	Read portion of RMW instruction may execute twice
G24	X	X	X	X	X	X	X	X		NoFix	MC2_STATUS MSR has model-specific error code and Machine Check Architecture error code reversed
G25	X	X	X	X	X	X	X	X		NoFix	MOVD, CVTSI2SS, or PINSRW following zeroing instruction can cause incorrect result
G26	X	X	X	X	X	X	X	X		NoFix	Top 4 PAT entries not usable with Mode B or Mode C paging
G27	X	X	X	X	X	X	X	X		NoFix	MOV with debug register causes debug exception
G28	X	X	X	X	X	X	X	X		NoFix	Data breakpoint exception in a displacement relative near call may corrupt EIP
G29	X	X	X	X	X	X	X	X		NoFix	System bus ECC not functional with 2:1 ratio
G30	X	X	X	X	X	X	X	X		NoFix	RDMSR or WRMSR to invalid MSR address may not cause GP fault
G31	X	X	X	X	X	X	X	X		NoFix	SYSENTER/SYSEXIT instructions can implicitly load “null segment selector” to SS and CS registers
G32	X	X	X	X	X	X	X	X		NoFix	PRELOAD followed by EXTEST does not load boundary scan data
G33	X	X	X							Fixed	Far jump to new TSS with D-bit cleared may cause system hang
G34	X	X								Fixed	Illegal opcode during L2 cache initialization
G35	X									Fixed	Incorrect L2 cache line invalidation
G36	X									Fixed	Transmission error on cache read
G37	X									Fixed	COMISS/UCOMISS may not update EFLAGS under certain conditions
G38	X	X	X	X	X	X	X	X		NoFix	System hang may occur with 2:1 core to bus ratio
G39	X	X	X	X	X	X	X	X		NoFix	Misaligned locked access to APIC space results in hang
G40	X	X	X	X	X	X	X	X		NoFix	Potential loss of data coherency during MP data ownership transfer
G41	X	X	X	X	X	X	X	X		NoFix	INT 1 instruction handler execution could generate a debug exception
G42	X	X	X	X	X	X	X	X		NoFix	Memory ordering-based synchronization may cause a livelock condition in MP systems
G43	X	X	X			X	X			Fixed	Floating-point exception signal may be deferred
G44	X									Fixed	System bus address parity checking may report false AERR#
G45	X	X	X	X	X	X	X	X		NoFix	Processor may assert DRDY# on a write with no data

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B 0	C 0	A 2	B 0	C 0	A 0	A 1	B 0			
G46									X	Fixed	Thermal sensor leakage current may exceed specification
G47	X	X	X	X	X	X	X	X		NoFix	GP# fault on WRMSR to ROB_CR_BKUPTMPDR6
G48									X	NoFix	Heavy L2 cache traffic results in noise on the TCK signal
G49	X	X	X	X		X	X			Fixed	Machine check exception may occur due to improper line eviction in the IFU
G50	X	X								Fixed	Machine check exception may occur during L2 cache initialization.
G51	X	X	X	X	X	X	X	X		NoFix	Snoop request may cause DBSY# hang
G52	X	X	X			X	X			Fixed	Performance counters include Streaming SIMD Extensions L1 prefetch
G53	X	X	X	X	X	X	X	X		NoFix	Lower bits of SMRAM SMBASE register cannot be written with an ITP
G54	X	X	X			X	X			Fixed	Task switch may cause wrong PTE and PDE access bit to be set
G55	X	X	X	X	X	X	X	X		NoFix	Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results
G56									X	NoFix	V_{IH} specification deviation on processor RS0# input
G57			X			X				Fixed	Noise sensitivity issue on processor SMI# pin
G58			X			X	X			Fixed	Processor will erroneously report a BIST failure
G59			X							Fixed	L2_LD and L2_M_LINES_OUTM performance-monitoring counters do not work
G60			X			X	X			Fixed	Limitation on cache line ECC detection and correction
G61			X			X	X			Fixed	IFU/DCU deadlock may cause system hang
G62			X			X	X			Fixed	L2_DBUS_BUSY performance monitoring counter will not count writes
G63	X	X	X	X		X	X			Fixed	Deadlock may occur due to illegal-instruction/page-miss combination
G64	X	X								Fixed	Incorrect sign may occur on X87 result due to indefinite QnaN result from Streaming SIMD Extensions multiply
G65	X	X	X	X	X	X	X			Fixed	MASKMOVQ instruction interaction with string operation may cause livelock
G66	X	X	X	X	X	X	X	X		NoFix	FLUSH# assertion following STPCLK# may prevent CPU clocks from stopping

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B 0	C 0	A 2	B 0	C 0	A 0	A 1	B 0			
G67	X	X	X	X	X	X	X	X		NoFix	Floating-point exception condition may be deferred
G68			X							Fixed	Intermittent failure to assert ADS# during system power-on
G69								X	X	NoFix	Race conditions may exist on thermal sensor SMBus collision detection/arbitration circuitry
G70			X	X		X				Fixed	Cache line reads may result in eviction of invalid data
G71						X				Fixed	Receiver noise sensitivity may result in system bus single-bit ECC errors
G72						X				Fixed	AGTL+ receiver may induce falling edge ledges and undershoot levels
G73	X	X	X	X	X	X	X	X		NoFix	Snoop probe during FLUSH# could cause L2 to be left in shared state
G74	X	X	X	X		X				Fixed	Livelock may occur due to IFU line eviction
G75						X	X			Fixed	L2 cache may not be correctly initialized following a power-on reset
G76			X	X		X	X			Fixed	Selector for the LTR/LLDT register may get corrupted
G77	X	X	X	X	X	X	X	X		NoFix	INIT does not clear global entries in the TLB
G78	X	X	X	X	X	X	X	X		NoFix	VM bit will be cleared on a double fault handler
G79	X	X	X	X	X	X	X	X		NoFix	Memory aliasing with inconsistent A and D bits may cause processor deadlock
G80	X	X	X	X	X	X	X	X		NoFix	Use of memory aliasing with inconsistent memory type may cause system hang
G81	X	X	X	X	X	X	X	X		NoFix	Processor may report invalid TSS fault instead of double fault during mode C paging
G82	X	X	X	X	X	X	X	X		NoFix	Machine check exception may occur when interleaving code between different memory types
G83	X	X	X	X	X	X	X	X		NoFix	Wrong ESP register values during a fault in VM86 mode
G84	X	X	X	X	X	X	X	X		NoFix	APIC ICR write may cause interrupt not to be sent when ICR delivery bit pending
G85				X						Fixed	High temperature and low supply voltage operation may result in incorrect processor operation
G86	X	X	X	X	X	X	X	X		NoFix	The Instruction Fetch Unit (IFU) may fetch instructions based upon stale CR3 data after a write to CR3 register
G87	X	X	X	X	X	X	X	X		NoFix	Under some complex conditions, the instructions in the shadow of a JMP FAR may be unintentionally executed and retired

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B 0	C 0	A 2	B 0	C 0	A 0	A 1	B 0			
G88	X	X	X	X	X	X	X	X		NoFix	Exx. Processor Does not Flag #GP on Non-zero Write to Certain MSRs
G89	X	X	X	X	X	X	X	X		NoFix	POPF and POPFD instructions that set the Trap Flag bit may cause unpredictable behavior
G90	X	X	X	X	X	X	X	X		NoFix	FXSAVE after FNINIT without an intervening FP (Floating Point) instruction may save unititialized values for FDP (x87 FPU Instruction Operand (Data) Pointer Offset and FDS (x87 FPU Instruction Operand (Data) Pointer Selector)
G91	X	X	X	X	X	X	X	X		NoFix	FSTP (Floating Point Store) instruction under certain conditions may result in erroneously setting a valid bit on an FP (Floating Point) stack register
G92	X	X	X	X	X	X	X	X		NoFix	Page with PAT (Page Attribute Table) set to USWC (Uncacheable Speculative Write Combine) while associated MTRR (Memory Type Range Register) is UC (Uncacheable) may consolidate to UC
G93	X	X	X	X	X	X	X	X		NoFix	Under certain conditions LTR (Load Task Register) instruction may result in system hang
G94	X	X	X	X	X	X	X	X		NoFix	Load from memory type USWC (Uncacheable Speculative Write Combine) may get its data forwarded from a previous pending store
G95	X	X	X	X	X	X	X	X		NoFix	Code Segment limit violation may occur on 4 Gigabyte limit check
G96	X	X	X	X	X	X	X	X		NoFix	FST instruction with numeric and null segment exceptions may cause General Protection Faults to be missed and FP Linear Address (FLA) to mismatch
G97	X	X	X	X	X	X	X	X		NoFix	Code Segment (CS) is incorrect on SMM handler when SMBASE is not aligned
G1AP	X	X	X		X	X	X	X		NoFix	APIC access to cacheable memory causes shutdown
G2AP	X	X	X		X	X	X	X		NoFix	MP systems may hang due to catastrophic errors during BSP determination
G3AP	X	X	X		X	X	X	X		NoFix	Write to mask LVT (programmed as EXTINT) will not deassert outstanding interrupt
G4AP	X	X	X	X	X	X	X	X		NoFix	REP MOVS Operation in Fast string Mode Continues in that Mode When Crossing into a Page with a Different Memory Type

Summary of Errata

NO.	Processor Signature/Stepping								PKG	Plans	ERRATA
	67*x*h		68*x*h			6A*x*h					
	B0	C0	A2	B0	C0	A0	A1	B0			
G5AP	X	X	X	X	X	X	X	X		NoFix	The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page with a UC Memory Type

NOTE: The erratum titled *A20M# may be inverted after returning from SMM and Reset* and numbered as G15 has been removed. It was erroneously included in previous versions of this Specification Update. The remaining errata have been renumbered.

Summary of Documentation Changes

NO.	CPUID/Stepping								PKG	Plans	DOCUMENTATION CHANGES
	67*x*h		68*x*h			6A*x*h					
	B0	C0	A2	B0	C0	A0	A1	B0			

Summary of Specification Clarifications

NO.	CPUID/Stepping								PKG	Plans	SPECIFICATION CLARIFICATONS
	67*x*h		68*x*h			6A*x*h					
	B0	C0	A2	B0	C0	A0	A1	B0			
G1	X	X	X	X	X	X	X	X		Doc	Specification clarification with respect to time stamp counter

Summary of Specification Changes

NO.	CPUID/Stepping								PKG	Plans	SPECIFICATION CHANGES
	67*x*h		68*x*h			6A*x*h					
	B0	C0	A2	B0	C0	A0	A1	B0			

ERRATA

G1. FP Data Operand Pointer May Be Incorrectly Calculated After FP Access Which Wraps 64-Kbyte Boundary in 16-Bit Code

Problem: The FP Data Operand Pointer is the effective address of the operand associated with the last noncontrol floating-point instruction executed by the machine. If an 80-bit floating-point access (load or store) occurs in a 16-bit mode other than protected mode (in which case the access will produce a segment limit violation), the memory access wraps a 64-Kbyte boundary, and the floating-point environment is subsequently saved, the value contained in the FP Data Operand Pointer may be incorrect.

Implication: A 32-bit operating system running 16-bit floating-point code may encounter this erratum, under the following conditions:

- The operating system is using a segment greater than 64 Kbytes in size.
- An application is running in a 16-bit mode other than protected mode.
- An 80-bit floating-point load or store which wraps the 64-Kbyte boundary is executed.
- The operating system performs a floating-point environment store (FSAVE/FNSAVE/FSTENV/FNSTENV) after the above memory access.
- The operating system uses the value contained in the FP Data Operand Pointer.

Wrapping an 80-bit floating-point load around a segment boundary in this way is not a normal programming practice. Intel has not currently identified any software which exhibits this behavior.

Workaround: If the FP Data Operand Pointer is used in an OS which may run 16-bit floating-point code, care must be taken to ensure that no 80-bit floating-point accesses are wrapped around a 64-Kbyte boundary.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G2. Differences Exist in Debug Exception Reporting

Problem: There exist some differences in the reporting of code and data breakpoint matches between that specified by previous Intel processor specifications and the behavior of the Pentium III Xeon processor, as described below:

Case 1: The first case is for a breakpoint set on a MOVSS or POPSS instruction, when the instruction following it causes a debug register protection fault (DR7.gd is already set, enabling the fault). The processor reports delayed data breakpoint matches from the MOVSS or POPSS instructions by setting the matching DR6.bi bits, along with the debug register protection fault (DR6.bd). If additional breakpoint faults are matched during the call of the debug fault handler, the processor sets the breakpoint match bits (DR6.bi) to reflect the breakpoints matched by both the MOVSS or POPSS breakpoint and the debug fault handler call. The Pentium III Xeon processor only sets DR6.bd in either situation, and does not set any of the DR6.bi bits.

Case 2: In the second breakpoint reporting failure case, if a MOVSS or POPSS instruction with a data breakpoint is followed by a store to memory which:

a) crosses a 4-Kbyte page boundary,

OR

b) causes the page table Access or Dirty (A/D) bits to be modified,

the breakpoint information for the MOVSS or POPSS will be lost. Previous processors retain this information under these boundary conditions.

Case 3: If they occur after a MOVSS or POPSS instruction, the INTn, INTO, and INT3 instructions zero the DR6.Bi bits (bits B0 through B3), clearing pending breakpoint information, unlike previous processors.

Case 4: If a data breakpoint and an SMI (System Management Interrupt) occur simultaneously, the SMI will be serviced via a call to the SMM handler, and the pending breakpoint will be lost.

Case 5: When an instruction that accesses a debug register is executed, and a breakpoint is encountered on the instruction, the breakpoint is reported twice.

Case 6: Unlike previous versions of Intel Architecture processors P6 family processors will not set the Bi bits for a matching disabled breakpoint unless at least one other breakpoint is enabled.

Implication: When debugging or when developing debuggers for a Pentium III Xeon processor-based system, this behavior should be noted. Normal usage of the MOVSS or POPSS instructions (i.e., following them with a MOV ESP) will not exhibit the behavior of cases 1-3. Debugging in conjunction with SMM will be limited by case 4.

Workaround: Following MOVSS and POPSS instructions with a MOV ESP instruction when using breakpoints will avoid the first three cases of this erratum. No workaround has been identified for cases 4, 5, or 6.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G3. FLUSH# Servicing Delayed While Waiting for STARTUP_IPI in MP Systems

Problem: In an MP system, if an application processor is waiting for a startup inter-processor interrupt (STARTUP_IPI), then it will not service a FLUSH# pin assertion until it has received the STARTUP_IPI.

Implication: After the MP initialization protocol, only one processor becomes the bootstrap processor (BSP). The other processor becomes a slave application processor (AP). After losing the BSP arbitration, the AP goes into a wait loop, waiting for a STARTUP_IPI.

The BSP can wake up the AP to perform some tasks with a STARTUP_IPI, and then put it back to sleep with an initialization inter-processor interrupt (INIT_IPI, which has the same effect as asserting INIT#), which returns it to a wait loop. The result is a possible loss of cache coherency if the off-line processor is intended to service a FLUSH# assertion at this point. The FLUSH# will be serviced as soon as the processor is awakened by a STARTUP_IPI, before any other instructions are executed. Intel has not encountered any operating systems that are affected by this erratum.

Workaround: Operating system developers should take care to execute a WBINVD instruction before the AP is taken off-line using an INIT_IPI.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G4. Code Fetch Matching Disabled Debug Register May Cause Debug Exception

Problem: The bits L0-3 and G0-3 enable breakpoints local to a task and global to all tasks, respectively. If one of these bits is set, a breakpoint is enabled, corresponding to the addresses in the debug registers DR0-DR3. If at least one of these breakpoints is enabled, any of these registers are *disabled* (i.e., L*n* and G*n* are 0), and RW*n* for the disabled register is 00 (indicating a breakpoint on instruction execution), normally an instruction fetch will not cause an instruction-breakpoint fault based on a match with the address in the disabled register(s). However, if the address in a disabled register matches the address of a code fetch which also results in a page fault, an instruction-breakpoint fault will occur.

Implication: While debugging software, extraneous instruction-breakpoint faults may be encountered if breakpoint registers are not cleared when they are disabled. Debug software which does not implement a code breakpoint handler will fail, if this occurs. If a handler is present, the fault will be serviced. Mixing data and code may exacerbate this problem by allowing disabled data breakpoint registers to break on an instruction fetch.

Workaround: The debug handler should clear breakpoint registers before they become disabled.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G5. Double ECC Error on Read May Result in BINIT#

Problem: For this erratum to occur, the following conditions must be met:

- Machine Check Exceptions (MCEs) must be enabled.
- A dataless transaction (such as a write invalidate) must be occurring simultaneously with a transaction which returns data (a normal read).
- The read data must contain a double-bit uncorrectable ECC error.

If these conditions are met, the Pentium III Xeon processor will not be able to determine which transaction was erroneous, and instead of generating an MCE, it will generate a BINIT#.

Implication: The bus will be reinitialized in this case. However, since a double-bit uncorrectable ECC error occurred on the read, the MCE handler (which is normally reached on a double-bit uncorrectable ECC error for a read) would most likely cause the same BINIT# event.

Workaround: Though the ability to drive BINIT# can be disabled in the Pentium III Xeon processor, which would prevent the effects of this erratum, overall system behavior would not improve, since the error which would normally cause a BINIT# would instead cause the machine to shut down. No other workaround has been identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G6. FP Inexact-Result Exception Flag May Not Be Set

Problem: When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real
- FIST m16int
- FIST m32int
- FISTP m16int
- FISTP m32int
- FISTP m64int

Note that even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.

Implication: Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a “sticky” bit, i.e., once set by an inexact-result condition, it remains set until cleared by software.

Workaround: This condition can be avoided by inserting two NOP instructions between the two floating-point instructions.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G7. BTM for SMI Will Contain Incorrect FROM EIP

Problem: A system management interrupt (SMI) will produce a Branch Trace Message (BTM), if BTMs are enabled. However, the FROM EIP field of the BTM (used to determine the address of the instruction which was being executed when the SMI was serviced) will not have been updated for the SMI, so the field will report the same FROM EIP as the previous BTM.

Implication: A BTM which is issued for an SMI will not contain the correct FROM EIP, limiting the usefulness of BTMs for debugging software in conjunction with System Management Mode (SMM).

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G8. I/O Restart in SMM May Fail After Simultaneous MCE

Problem: If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the Pentium III Xeon processor will signal a machine check exception (MCE). If the instruction is directed at a device which is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.

Implication: A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler may attempt to restart such an I/O instruction, but will have corrupted state due to the MCE handler call, leading to failure of the restart and shutdown of the processor.

Workaround: If a system implementation must support both SMM and MCEs, the first thing the SMM handler code (when an I/O restart is to be performed) should do is check for a pending MCE. If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the machine check exception handler to execute. If there is not, the SMM handler may proceed with its normal operation.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G9. Branch Traps Do Not Function if BTMs Are Also Enabled

Problem: If branch traps or branch trace messages (BTMs) are enabled alone, both function as expected. However, if both are enabled, only the BTMs will function, and the branch traps will be ignored.

Implication: The branch traps and branch trace message debugging features cannot be used together.

Workaround: If branch trap functionality is desired, BTMs must be disabled.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G10. Checker BIST Failure in FRC Mode Not Signaled

Problem: If a system is running in functional redundancy checking (FRC) mode, and the checker of the master-checker pair encounters a hard failure while running the built-in self test (BIST), the checker will tri-state all outputs without signaling an IERR#.

Implication: Assuming the master passes BIST successfully, it will continue execution unchecked, operating without functional redundancy. However, the necessary pull-up on the FRCERR pin will cause an FRCERR to be signaled. The operation of the master depends on the implementation of FRCERR.

Workaround: For successful detection of BIST failure in the checker of an FRC pair, use the FRCERR signal, instead of IERR#.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G11. BINIT# Assertion Causes FRCERR Assertion in FRC Mode

Problem: If a pair of Pentium III Xeon processors are running in functional redundancy checking (FRC) mode, and a catastrophic error condition causes BINIT# to be asserted, the checker in the master-checker pair will enter shutdown. The next bus transaction from the master will then result in the assertion of FRCERR.

Implication: Bus initialization via an assertion of BINIT# occurs as the result of a catastrophic error condition which precludes the continuing reliable execution of the system. Under normal circumstances, the master-checker pair would remain synchronized in the execution of the BINIT# handler. However, due to this erratum, an FRCERR will be signaled. System behavior then depends on the system-specific, error recovery mechanisms.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G12. Machine Check Exception Handler May Not Always Execute Successfully

Problem: An asynchronous machine check exception (MCE), such as a BINIT# event, which occurs during an access that splits a 4-Kbyte page boundary may leave some internal registers in an indeterminate state. Thus, MCE handler code may not always run successfully if an asynchronous MCE has occurred previously.

Implication: An MCE may not always result in the successful execution of the MCE handler. However, asynchronous MCEs usually occur upon detection of a catastrophic system condition that would also hang the processor. Leaving MCEs disabled will result in the condition which caused the asynchronous MCE instead causing the processor to enter shutdown. Therefore, leaving MCEs disabled may not improve overall system behavior.

Workaround: No workaround which would guarantee successful MCE handler execution under this condition has been identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G13. LBER May Be Corrupted After Some Events

Problem: The last branch record (LBR) and the last branch before exception record (LBER) can be used to determine the source and destination information for previous branches or exceptions. The LBR contains the source and destination addresses for the last branch or exception, and the LBER contains similar information for the last branch taken before the last exception. This information is typically used to determine the location of a branch which leads to execution of code which causes an exception. However, after a catastrophic bus condition which results in an assertion of BINIT# and the re-initialization of the buses, the value in the LBER may be corrupted. Also, after either a CALL which results in a fault or a software interrupt, the LBER and LBR will be updated to the same value, when the LBER should not have been updated.

Implication: The LBER and LBR registers are used only for debugging purposes. When this erratum occurs, the LBER will not contain reliable address information. The value of LBER should be used with caution when debugging branching code; if the values in the LBR and LBER are the same, then the LBER value is incorrect. Also, the value in the LBER should not be relied upon after a BINIT# event.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G14. BTMs May Be Corrupted During Simultaneous L1 Cache Line Replacement

Problem: When Branch Trace Messages (BTMs) are enabled and such a message is generated, the BTM may be corrupted when issued to the bus by the L1 cache if a new line of data is brought into the L1 data cache simultaneously. Though the new line being stored in the L1 cache is stored correctly, and no corruption occurs in the data, the information in the BTM may be incorrect due to the internal collision of the data line and the BTM.

Implication: Although BTMs may not be entirely reliable due to this erratum, the conditions necessary for this boundary condition to occur have only been exhibited during focused simulation testing. Intel has currently not observed this erratum in a system level validation environment.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G15. Near CALL to ESP Creates Unexpected EIP Address

Problem: As documented, the CALL instruction saves procedure linking information in the procedure stack and jumps to the called procedure specified with the destination (target) operand. The target operand specifies the address of the first instruction in the called procedure. This operand can be an immediate value, a general purpose register, or a memory location. When accessing an absolute address indirectly using the stack pointer (ESP) as a base register, the base value used is the value in the ESP register before the instruction executes. However, when accessing an absolute address directly using ESP as the base register, the base value used is the value of ESP *after* the return value is pushed on the stack, not the value in the ESP register *before* the instruction executed.

Implication: Due to this erratum, the processor may transfer control to an unintended address. Results are unpredictable, depending on the particular application, and can range from no effect to the unexpected termination of the application due to an exception. Intel has observed this erratum only in a focused testing environment. Intel has not observed any commercially available operating system, application, or compiler that makes use of or generates this instruction.

Workaround: If the other seven general purpose registers are unavailable for use, and it is necessary to do a CALL via the ESP register, first push ESP onto the stack, then perform an *indirect* call using ESP (e.g., CALL [ESP]). The saved version of ESP should then be popped off the stack after the call returns.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G16. Mixed Cacheability of Lock Variables Is Problematic in MP Systems

Problem: This errata only affects multiprocessor systems where a lock variable address is marked cacheable in one processor and uncacheable in any others. The processors which have it marked uncacheable may stall indefinitely when accessing the lock variable. The stall is only encountered if:

- One processor has the lock variable cached, and is attempting to execute a cache lock.
- If the processor which has that address cached has it cached in its L2 only.
- Other processors, meanwhile, issue back to back accesses to that same address on the bus.

Implication: MP systems where all processors either use cache locks or consistent locks to uncacheable space will not encounter this problem. If, however, a lock variable's cacheability varies in different processors, and several processors are all attempting to perform the lock simultaneously, an indefinite stall may be experienced by the processors which have it marked uncacheable in locking the variable (if the conditions above are satisfied). Intel has only encountered this problem in focus testing with artificially generated external events. Intel has not currently identified any commercial software which exhibits this problem.

Workaround: Follow a homogenous model for the memory type range registers (MTRRs), ensuring that all processors have the same cacheability attributes for each region of memory; do not use locks whose memory type is cacheable on one processor, and uncacheable on others. Avoid page table aliasing, which may produce a nonhomogenous memory model.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G17. MCE Due to L2 Parity Error Gives L1 MCACOD.LL

Problem: If a Cache Reply Parity (CRP) error, Cache Address Parity (CAP) error, or Cache Synchronous Error (CSER) occurs on an access to the Pentium III Xeon processor's L2 cache, the resulting Machine Check Architectural Error Code (MCACOD) will be logged with '01' in the LL field. This value indicates an L1 cache error; the value should be '10', indicating an L2 cache error. Note that L2 ECC errors have the correct value of '10' logged.

Implication: An L2 cache access error, other than an ECC error, will be improperly logged as an L1 cache error in MCACOD.LL.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G18. Memory Type Field Undefined for Nonmemory Operations

Problem: The Memory Type field for nonmemory transactions such as I/O and Special Cycles are undefined. Although the Memory Type attribute for nonmemory operations logically should (and usually does) manifest itself as UC, this feature is not designed into the implementation and is therefore inconsistent.

Implication: Bus agents may decode a non-UC memory type for nonmemory bus transactions.

Workaround: Bus agents must consider transaction type to determine the validity of the Memory Type field for a transaction.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G19. Infinite Snoop Stall During L2 Initialization of MP Systems

Problem: It is possible for snoop traffic generated on the system bus while a processor is executing its L2 cache initialization routine to cause the initializing processor to hang.

Implication: An MP system which does not suppress snoop traffic while L2 caches are being initialized may hang during this initialization sequence.

Workaround: System BIOS can create an execution environment which allows processors to initialize their L2 caches without the system generating any snoop traffic on the bus.
Below is a pseudo-code fragment, designed explicitly for a four-processor system, that uses a serial algorithm to initialize each processor's L2 cache:

```
Suppress_all_I/O_traffic()
 K = 0;
 while (K <= 3)
 {
 /* Obtain current value of K. This forces both Temp and K into */
 /* the L1 cache. Note that Temp could also be maintained in a */
 /* general purpose register. */

 Temp = K;
 Wait_until_all_processors_are_signed_in_at_barrier()
 if ( logical_proc_APIC_id == K ) {
 {
 wait_10_usecs_delay_loop(); /* this time delay, required */
 /* in the worst case, allows */
 /* the barrier semaphore to */
 /* settle to shared state. */
 Initialize L2 cache
 K++
 }
 else
 while (Temp == K);
 }
 }
```

This algorithm prevents bus snoop traffic from the other processors, which would otherwise cause the initializing processor to hang. The algorithm assumes that the L1 cache is enabled (the Temp and K variables must be cached by each processor). Also, the Memory Type Range Register (MTRR) for the data segment must be set to WB (writeback) memory type.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G20. FP Data Operand Pointer May Not Be Zero After Power On or Reset

Problem: The FP Data Operand Pointer, as specified, should be reset to zero upon power on or Reset by the processor. Due to this erratum, the FP Data Operand Pointer may be nonzero after power on or Reset.

Implication: Software which uses the FP Data Operand Pointer and count on its value being zero after power on or Reset without first executing an FINIT/FNINIT instruction will use an incorrect value, resulting in incorrect behavior of the software.

Workaround: Software should follow the recommendation in Section 8.2 of the *Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide* (Order Number 243192). This recommendation states that if the FPU will be used, software-initialization code should execute an FINIT/FNINIT instruction following a hardware reset. This will correctly clear the FP Data Operand Pointer to zero.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G21. Premature Execution of Load Operation Prior to Exception Handler Invocation

Problem: This erratum can occur in any of the following situations:

1. If an instruction that performs a memory load causes a code segment limit violation,
2. If a waiting floating-point instruction or MMX™ instruction that performs a memory load has a floating-point exception pending, or
3. If an MMX instruction that performs a memory load and has either CR0.EM = 1 (Emulation bit set), a floating-point Top-of-Stack (FP TOS) not equal to 0, or a DNA exception pending.

If any of the above circumstances occur, it is possible that the load portion of the instruction will have executed before the exception handler is entered.

Implication: In normal code execution where the target of the load operation is to write back memory there is no impact from the load being prematurely executed, nor from the restart and subsequent re-execution of that instruction by the exception handler. If the target of the load is to uncached memory that has a system side-effect, restarting the instruction may cause unexpected system behavior due to the repetition of the side-effect.

Workaround: Code which performs loads from memory that has side-effects can effectively work around this behavior by using simple integer-based load instructions when accessing side-effect memory, and by ensuring that all code is written such that a code segment limit violation cannot occur as a part of reading from side-effect memory.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G22. EFLAGS Discrepancy on Page Fault After Multiprocessor TLB Shootdown

Problem: This erratum may occur when the Pentium III Xeon processor executes one of the following read-modify-write arithmetic instructions and a page fault occurs during the store of the memory operand: ADD, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT, OR, ROL/ROR, SAL/SAR/SHL/SHR, SHLD, SHRD, SUB, XOR, and XADD. In this case, the EFLAGS value pushed onto the stack of the page fault handler may reflect the status of the register after the instruction would have completed execution rather than before it. The following conditions are required for the store to generate a page fault and call the operating system page fault handler:

1. The store address entry must be evicted from the DTLB by speculative loads from other instructions that hit the same way of the DTLB before the store has completed. DTLB eviction requires at least three load

operations that have linear address bits 15:12 equal to each other and address bits 31:16 different from each other in close physical proximity to the arithmetic operation.

2. The page table entry for the store address must have its permissions tightened during the very small window of time between the DTLB eviction and execution of the store. Examples of page permission tightening include from Present to Not Present or from Read/Write to Read Only, etc.
3. Another processor, without corresponding synchronization and TLB flush, must cause the permission change.

Implication: This scenario may only occur on a multiprocessor platform running an operating system that performs “lazy” TLB shootdowns. The memory image of the EFLAGS register on the page fault handler’s stack prematurely contains the final arithmetic flag values although the instruction has not yet completed. Intel has not identified any operating systems that inspect the arithmetic portion of the EFLAGS register during a page fault nor observed this erratum in laboratory testing of software applications.

Workaround: No workaround is needed upon normal restart of the instruction, since this erratum is transparent to the faulting code and results in correct instruction behavior. Operating systems may ensure that no processor is currently accessing a page that is scheduled to have its page permissions tightened or have a page fault handler that ignores any incorrect state.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G23. Read Portion of RMW Instruction May Execute Twice

Problem: When the Pentium III Xeon processor executes a read-modify-write (RMW) arithmetic instruction, with memory as the destination, it is possible for a page fault to occur during the execution of the store on the memory operand after the read operation has completed but before the write operation completes. If the memory targeted for the instruction is UC (uncached), memory will observe the occurrence of the initial load before the page fault handler and again if the instruction is restarted.

Implication: If the memory targeted for the RMW instruction has no side effects, then the memory location will simply be read twice with no additional implications. If, however, the load targets a memory region that has side effects, multiple occurrences of the initial load may lead to unpredictable system behavior.

Workaround: Hardware and software developers who write device drivers for custom hardware that may have a side effect style of design should use simple loads and simple stores to transfer data to and from the device.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G24. MC2_STATUS MSR Has Model-Specific Error Code and Machine Check Architecture Error Code Reversed

Problem: The *Intel Architecture Software Developer’s Manual, Volume 3: System Programming Guide,* documents that for the MC*i*_STATUS MSR, bits 15:0 contain the MCA (machine-check architecture) error code field, and bits 31:16 contain the model-specific error code field. However, for the MC2_STATUS MSR, these bits have been reversed. For the MC2_STATUS MSR, bits 15:0 contain the model-specific error code field and bits 31:16 contain the MCA error code field.

Implication: A machine check error may be decoded incorrectly if this erratum on the MC2_STATUS MSR is not taken into account.

Workaround: When decoding the MC2_STATUS MSR, reverse the two error fields.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G25. *MOVD, CVTSI2SS, or PINSRW following zeroing instruction can cause incorrect result*

Problem: An incorrect result may be calculated after the following circumstances occur:
1. A register has been zeroed with either a SUB reg, reg instruction, or an XOR reg, reg instruction.
2. A value is moved with sign extension into the same register’s lower 16 bits; or a signed integer multiply is performed to the same register’s lower 16 bits.
3. The register is then copied to an MMX™ technology register using the MOVD, or converted to single precision floating-point and moved to an MMX technology register using the CVTSI2SS instruction prior to any other operations on the sign-extended value, or inserted into an MMX™ technology register using the PINSRW instruction.

Specifically, the sign may be incorrectly extended into bits 16-31 of the MMX technology register. In the case of the PINSRW instruction, a non-zero value could be loaded into the MMX™ technology register. This erratum only affects the MMX™ technology register.

This erratum only occurs when the following three steps occur in the order shown. This erratum may occur with up to 63 (39 for Pre-CPUID 0x6BX) intervening instructions that do not modify the sign-extended value between steps 2 and 3.

1. XOR EAX, EAX
   or SUB EAX, EAX
2. MOVSX AX, BL
   or MOVSX AX, byte ptr <memory address> or MOVSX AX, BX
   or MOVSX AX, word ptr <memory address> or IMUL BL (AX implicit, opcode F6 /5)
   or IMUL byte ptr <memory address> (AX implicit, opcode F6 /5) or IMUL AX, BX (opcode 0F AF /r)
   or IMUL AX, word ptr <memory address> (opcode 0F AF /r) or IMUL AX, BX, 16 (opcode 6B /r ib)
   or IMUL AX, word ptr <memory address>, 16 (opcode 6B /r ib) or IMUL AX, 8 (opcode 6B /r ib)
   or IMUL AX, BX, 1024 (opcode 69 /r iw)
   or IMUL AX, word ptr <memory address>, 1024 (opcode 69 /r iw)
   or IMUL AX, 1024 (opcode 69 /r iw) or CBW
3. MOVD MM0, EAX or CVTSI2SS MM0, EAX

Note that the values for immediate byte/words are merely representative (i.e., 8, 16, 1024) and that any value in the range for the size is affected. Also, note that this erratum may occur with “EAX” replaced with any 32-bit general-purpose register, and “AX” with the corresponding 16-bit version of that replacement. “BL” or “BX” can be replaced with any 8-bit or 16-bit general-purpose register. The CBW and IMUL (opcode F6 /5) instructions are specific to the EAX register only.

In the above example, EAX is forced to contain 0 by the XOR or SUB instructions. Since the four types of the MOVSX or IMUL instructions and the CBW instruction only modify bits 15:8 of EAX by sign extending the lower 8 bits of EAX, bits 31:16 of EAX should always contain 0. This implies that when MOVD or CVTSI2SS copies EAX to MM0, bits 31:16 of MM0 should also be 0. In certain scenarios, bits 31:16 of MM0 are not 0, but are replicas of bit 15 (the 16th bit) of AX. This is noticeable when the value in AX after the MOVSX, IMUL or CBW instruction is negative (i.e., bit 15 of AX is a 1).

When AX is positive (bit 15 of AX is 0), MOVD or CVTSI2SS will produce the correct answer. If AX is negative (bit 15 of AX is 1), MOVD or CVTSI2SS may produce the right answer or the wrong answer, depending on the point in time when the MOVD or CVTSI2SS instruction is executed in relation to the MOVSX, IMUL or CBW instruction.

The PINSRW instruction can fail to correctly load a zero when used with a partial register zeroing instruction (SUB or XOR):

1. mov di, 0FFFF8914h
2. xor eax, eax
3. add ax, di
4. xor ah, ah
5. pinsrw mm1, eax, 00h

In this case, the programmer expects mm1 to contain 0014h in it’s least significant word. This erratum would cause MM1 to contain 8914h. The number of intervening instructions between steps 4 and 5 is the same as noted in the sign extension example above between steps 2 and 3.

Implication: The effect of incorrect execution will vary from unnoticeable, due to the code sequence discarding the incorrect bits, to an application failure.

Workaround: There are two possible workarounds for this erratum:
1. Rather than using the MOVSX-MOVD/CVTSI2SS, IMUL-MOVD/CVTSI2SS or CBW-MOVD/CVTSI2SS pairing to handle one variable at a time, use the sign extension capabilities (PSRAW, etc.) within MMX technology for operating on multiple variables. This will also result in higher performance.
2. Insert another operation that modifies or copies the sign-extended value between the MOVSX/IMUL/CBW instruction and the MOVD or CVTSI2SS instruction as in the example below:

```
XOR EAX, EAX (or SUB EAX, EAX)
MOVSX AX, BL (or other MOVSX, other IMUL or CBW instruction)
*MOV EAX, EAX
MOVD MM0, EAX or CVTSI2SS MM0, EAX
```

3. Avoid using a sub or xor to zero a partial register prior to the use of any of these three instructions. Instead, use a mov immediate (e.g. “mov ah, 0h”).
*Note: MOV EAX, EAX is used here in a generic sense. Again, EAX can be substituted with any 32-bit register.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G26. Top 4 PAT Entries Not Usable With Mode B or Mode C Paging

Problem: The Page Attribute Table (PAT) contains eight entries, which must all be initialized and considered when setting up memory types for the Pentium III Xeon processor. However, in Mode B or Mode C paging, the top four entries do not function correctly for 4-Kbyte pages. Specifically, bit 7 of page table entries which translate addresses to 4-Kbyte pages should be used as the upper bit of a three-bit index to determine the PAT entry that specifies the memory type for the page. When Mode B (CR4.PSE = 1) and/or Mode C (CR4.PAE = 1) are enabled, the processor forces this bit to zero when determining the memory type, regardless of the value in the page table entry. The upper four entries of the PAT function correctly for 2-Mbyte and 4-Mbyte large pages (specified by bit 12 of the page directory entry for those translations).

Implication: Only the lower four PAT entries are useful for 4-Kbyte translations when Mode B or C paging is used. In Mode A paging (4-Kbyte pages only), all eight entries may be used. All eight entries may also be used for large pages in Mode B or C paging.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G27. MOV with Debug Register Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed on debug registers, a general-protection exception (#GP) should be generated, as documented in the *Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide*, Section 14.2. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

Implication: With debug-register protection enabled (i.e., the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

Workaround: In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G28. Data Breakpoint Exception in a Displacement Relative Near Call May Corrupt EIP

Problem: If a misaligned data breakpoint is programmed to the same cache line as the memory location where the stack push of a near call is performed and any data breakpoints are enabled, the processor will update the stack and ESP appropriately, but may skip the code at the destination of the call. Hence, program execution will continue with the next instruction immediately following the call, instead of the target of the call.

Implication: The failure mechanism for this erratum is that the call would not be taken; therefore, instructions in the called subroutine would not be executed. As a result, any code relying on the execution of the subroutine will behave unpredictably.

Workaround: Whether enabled or not, do not program a misaligned data breakpoint to the same cache line on the stack where the push for the near call is performed.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section**.**

G29. System Bus ECC Not Functional With 2:1 Ratio

Problem: If a processor is underclocked at a core frequency to system bus frequency ratio of 2:1 and system bus ECC is enabled, the system bus ECC detection and correction will negatively affect internal timing dependencies.

Implication: If system bus ECC is enabled, and the processor is underclocked at a 2:1 ratio, the system may behave unpredictably due to these timing dependencies.

Workaround: All bus agents that support system bus ECC must disable it when a 2:1 ratio is used.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G30. RDMSR or WRMSR to Invalid MSR Address May Not Cause GP Fault

Problem: The RDMSR and WRMSR instructions allow reading or writing of MSRs (Model Specific Registers) based on the index number placed in ECX. The processor should reject access to any reserved or unimplemented MSRs by generating #GP(0). However, there are some invalid MSR addresses for which the processor will not generate #GP(0).

Implication: For RDMSR, undefined values will be read into EDX:EAX. For WRMSR, undefined processor behavior may result.

Workaround: Do not use invalid MSR addresses with RDMSR or WRMSR.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G31. SYSENTER/SYSEXIT Instructions Can Implicitly Load "Null Segment Selector" to SS and CS Registers

Problem: According to the processor specification, attempting to load a null segment selector into the CS and SS segment registers should generate a General Protection Fault (#GP). Although loading a null segment selector to the other segment registers is allowed, the processor will generate an exception when the segment register holding a null selector is used to access memory.

However, the SYSENTER instruction can implicitly load a null value to the SS segment selector. This can occur if the value in SYSENTER_CS_MSR is between FFF8h and FFFBh when the SYSENTER instruction is executed. This behavior is part of the SYSENTER/SYSEXIT instruction definition; the content of the SYSTEM_CS_MSR is always incremented by 8 before it is loaded into the SS. This operation will set the null bit in the segment selector if a null result is generated, but it does not generate a #GP on the SYSENTER instruction itself. An exception will be generated as expected when the SS register is used to access memory, however.

The SYSEXIT instruction will also exhibit this behavior for both CS and SS when executed with the value in SYSENTER_CS_MSR between FFF0h and FFF3h, or between FFE8h and FFEBh, inclusive.

Implication: These instructions are intended for operating system use. If this erratum occurs (and the OS does not ensure that the processor never has a null segment selector in the SS or CS segment registers), the processor's behavior may become unpredictable, possibly resulting in system failure.

Workaround: Do not initialize the SYSTEM_CS_MSR with the values between FFF8h and FFFBh, FFF0h and FFF3h, or FFE8h and FFEBh before executing SYSENTER or SYSEXIT.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G32. PRELOAD Followed by EXTEST Does Not Load Boundary Scan Data

Problem: According to the IEEE 1149.1 Standard, the EXTEST instruction would use data "typically loaded onto the latched parallel outputs of boundary-scan shift-register stages using the SAMPLE/PRELOAD instruction prior to the selection of the EXTEST instruction." As a result of this erratum, this method cannot be used to load the data onto the outputs.

Implication: Using the PRELOAD instruction prior to the EXTEST instruction will not produce expected data after the completion of EXTEST.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G33. Far Jump to New TSS With D-bit Cleared May Cause System Hang

Problem: A task switch may be performed by executing a far jump through a task gate or to a new Task State Segment (TSS) directly. Normally, when such a jump to a new TSS occurs, the D-bit (which indicates that the page referenced by a Page Table Entry (PTE) has been modified) for the PTE which maps the location of the previous TSS will already be set and the processor will operate as expected. However, if the D-bit is clear at the time of the jump to the new TSS, the processor will hang.

Implication: If an OS is used which can clear the D-bit for system pages, and which jumps to a new TSS on a task switch, then a condition may occur which results in a system hang. Intel has not identified any commercial software which may encounter this condition; this erratum was discovered in a focused testing environment.

Workaround: Ensure that OS code does not clear the D-bit for system pages (including any pages that contain a task gate or TSS). Use task gates rather than jumping to a new TSS when performing a task switch.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G34. Illegal Opcode During L2 Cache Initialization

Problem: It is possible for the cache components in the 1-Mbyte and 2-Mbyte Pentium III Xeon processor to power up in a state such that they are not synchronized. During a read under these circumstances, the data in the cache is correct but the processor does not read the data correctly.

Implication: The processor may read invalid data after the cache is enabled during the Power-On Self Test (POST) phase of boot-up, most likely resulting in an invalid opcode being received by the processor, which would generate an invalid opcode exception.

Workaround: Intel recommends that the following BIOS instructions (or equivalent) be added to the Intel L2 Cache initialization module, just prior to enabling the L2 cache via BBL_CR_CTL3 [8]:

```
MOV         ECX, 11Eh             ; MSR (11Eh) is BBL_CR_CTL3
RDMSR                             ; read contents
PUSH        EAX                   ; save lower 32 bits
PUSH        EDX                   ; save upper 32 bits
AND         AL, 0E1h              ; isolate latency bits
OR          AL, 00Ah              ; set to a desktop latency value
WRMSR                             ; write new value out
POP         EDX                   ; restore original value determined
POP         EAX                   ; by the BIOS for latency
WRMSR                             ; write it back out
```

IMPORTANT NOTE

The above example code contains stack operations. If the BIOS L2 cache initialization code is executed in a pre-stack environment, the BIOS developer must ensure that the push/pop instruction pairs are replaced with another register save method. Also, the BIOS developer must ensure that the actual BIOS code does not corrupt existing code's register usage.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G35. Incorrect L2 Cache Line Invalidation

Problem: In the event of a complex set of internal conditions the processor may invalidate an L2 cache line incorrectly. If this line was previously in the modified state, cache coherency is not maintained. Since timing is critical for these conditions to occur, the following sequence of events MUST occur in order to encounter this erratum:

1. An external snoop occurs.
2. An L1 cache read miss occurs which generates an L2 cache read.
3. This L2 cache read is a miss, causing a modified cache line in the L2 to be scheduled for eviction.
4. An L1 cache write-back (resulting from a separate transaction that happened before event 1) of a different cache line occurs, which causes a second L2 cache miss. The L2 set address is the same as the set address in the first external snoop.
5. Another external snoop occurs.
6. Another L2 cache read occurs; this read may result in either a hit or a miss.
7. The modified cache line in event 3 is written back to main memory, but is not yet invalidated.

The delay between event 3 and event 7 causes the processor to invalidate the wrong L2 cache line.

Implication: Stale data may exist in main memory because the newer, modified data was not written back. Depending on the significance of this modified cache line, results are unpredictable.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G36. Transmission Error on Cache Read

Problem: During reads of the L2 cache, the processor may use L2 cache optimizations which result in a data transmission error.

Implication: Data corruption caused by this erratum will result in unpredictable system behavior.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G37. COMISS/UCOMISS May Not Update EFLAGS Under Certain Conditions

Problem: COMISS/UCOMISS instructions compare the least significant pairs of packed single-precision floating-point numbers and set the ZF, PF and CF bits in the EFLAGS register accordingly (the OF, SF and AF bits are cleared). The EFLAGS register may not contain the appropriate value for the specified COMISS/UCOMISS operands under the following conditions:

1. The source operand of the COMISS/UCOMISS instruction is from memory that encounters some latency before it is available to the processor

   AND

2. An instruction subsequent to the COMISS/UCOMISS is a string instruction (e.g., REP MOVS) which completes before the UCOMISS/COMISS instruction completes.

Implication: The COMISS/UCOMISS instructions with a register source operand are unaffected by this erratum. The result of the incorrect status of the EFLAGS register may range from no effect to unexpected application or operating system behavior.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G38. System Hang May Occur With 2:1 Core to Bus Ratio

Problem: The following conditions must occur for this erratum to be observed:

1. Core frequency to bus frequency ratio must be 2:1.
2. The Chipset must use the most aggressive TRDY assertion allowed by the protocol.
3. Use one of the following instructions: movntq, movntps, or maskmovq.
   OR

   Do a large number of back to back stores to WC memory.

Implication: The occurrence of this erratum may cause a system hang.

Workaround: Do not use 2:1 core to bus ratio.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G39. Misaligned Locked Access to APIC Space Results in Hang

Problem: When the processor's APIC space is accessed with a misaligned locked access a machine check exception is expected. However, the processor's machine check architecture is unable to handle the misaligned locked access.

Implication: If this erratum occurs the processor will hang. Typical usage models for the APIC address space do not use locked accesses. Systems using such a model will not be affected by this erratum.

Workaround: Ensure that all accesses to APIC space are aligned and/or not locked.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G40. Potential Loss of Data Coherency During MP Data Ownership Transfer

Problem: In MP systems, processors may be sharing data in different cache lines, referenced as line A and line B in the discussion below. When this erratum occurs (with the following example given for a 2-way MP system with processors noted as 'P0' and 'P1'), P0 contains a shared copy of line B in its L1. P1 has a shared copy of Line A. Each processor must manage the necessary invalidation and snoop cycles before that processor can modify and source the results of any internal writes to the other processor.

There exists a narrow timing window when, if P1 requests a copy of line B it may be supplied by P0 in an Exclusive state which allows P1 to modify the contents of the line with no further external invalidation cycles. In this narrow window P0 may also retire instructions that use the original data present before P1 performed the modification.

Implication: Multiprocessor or threaded application synchronization, required for low-level data sharing, that is implemented via operating system provided synchronization constructs are not affected by this erratum. Applications that rely upon the usage of locked semaphores rather than memory ordering are also unaffected. Uniprocessor systems are not affected by this erratum. If the erratum does occur one processor may execute software with the stale data that was present from the previous shared state rather than the data written more recently by another processor.

Workaround: Deterministic barriers beyond which program variables will not be modified can be achieved via the usage of locked semaphore operations. These should effectively prevent the occurrence of this erratum.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G41. INT 1 Instruction Handler Execution Could Generate a Debug Exception

Problem: If the processor's general detect enable flag is set and an explicit call is made to the interrupt procedure via the INT 1 instruction, the general detect enable flag should be cleared prior to entering the handler. As a result of this erratum, the flag is not cleared prior to entering the handler. If an access is made to the debug registers while inside of the handler, the state of the general detect enable flag will cause a second debug exception to be taken. The second debug exception clears the general detect enable flag and returns control to the handler which is now able to access the debug registers.

Implication: This erratum will generate an unexpected debug exception upon accessing the debug registers while inside of the INT 1 handler.

Workaround: Ignore the second debug exception that is taken as a result of this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G42. Memory Ordering-Based Synchronization May Cause a Livelock Condition in MP Systems

Problem: In an MP environment, the following sequence of code (or similar code) in two processors (P0 and P1) may cause them to each enter an infinite loop (livelock condition):

	P0			P1		
		MOV [xyz], EAX	(1)	wait1:	MOV EBX, [abc]	(2)
		.			CMP EBX, val1	(3)
		.			JNE wait1	(4)
		.				
		MOV [abc], val1	(6)		MOV [abc], val2	(5)
wait0:		MOV EBX, [abc]	(7)			
		CMP EBX, val2	(8)			
		JNE wait0	(9)			

NOTE

EAX and EBX can be any general-purpose register. Addresses [abc] and [xyz] can be any location in memory and must be in the same bank of the L1 cache. Variables “val1” and “val2” can be any integer.

The algorithm above involves processors P0 and P1, each of which use loops to keep them synchronized with each other. P1 is looping until instruction (6) in P0 is globally observed. Likewise, P0 will loop until instruction (5) in P1 is globally observed.

The P6 architecture allows for instructions (1) and (7) in P0 to be dispatched to the L1 cache simultaneously. If the two instructions are accessing the same memory bank in the L1 cache, the load (7) will be given higher priority and will complete, blocking instruction (1).

Instructions (8) and (9) may then execute and retire, placing the instruction pointer back to instruction (7). This is due to the condition at the end of the “wait0” loop being false. The livelock scenario can occur if the timing of the wait0 loop execution is such that instruction (7) in P0 is ready for completion every time that instruction (1) tries to complete. Instruction (7) will again have higher priority, preventing the data ([xyz]) in instruction (1) from being written to the L1 cache. This causes instruction (6) in P0 to not complete and the sequence “wait0” to loop infinitely in P0.

A livelock condition also occurs in P1 because instruction (6) in P0 does not complete (blocked by instruction (1) not completing). The problem with this scenario is that P0 should eventually allow for instruction (1) to write its data to the L1 cache. If this occurs, the data in instruction (6) will be written to memory, allowing the conditions in both loops to be true.

Implication: Both processors will be stuck in an infinite loop, leading to a hang condition. Note that if P0 receives any interrupt, the loop timing will be disrupted such that the livelock will be broken. The system timer, a keystroke, or mouse movement can provide an interrupt that will break the livelock.

Workaround: Use a LOCK instruction to force P0 to execute instruction (6) before instruction (7).

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G43. Floating-Point Exception Signal May Be Deferred

Problem: A one clock window exists where a pending x87 FP exception that should be signaled on the execution of a CVTPS2PI, CVTPI2PS, or CVTTPS2PI instruction may be deferred to the next waiting floating-point instruction or instruction that would change MMX™ register state.

Implication: If this erratum occurs the floating-point exception will not be handled as expected.

Workaround: Applications that follow Intel programming guidelines (empty all x87 registers before executing MMX technology instructions) will not be affected by this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G44. System Bus Address Parity Checking May Report False AERR#

Problem: The processor's address parity error detection circuit may fail to meet its frequency timing specification under certain environmental conditions. At the high end of the temperature specification and/or the low end of the voltage range, the processor may report false address parity errors.

Implication: If the system has AERR# drive enabled (bit [3] of the EBL_CR_POWERON resister set to '1') spurious address detection and reporting may occur. In some system configurations BINIT# may be asserted on the system bus. This may cause some systems to generate a machine check exception and in others may cause a reboot.

Workaround: Disable AERR# drive from the processor. AERR# drive may be disabled by clearing bit [3] in the EBL_CR_POWERON register. In addition, if the chipset allows, AERR# drive should be enabled from the chipset and AERR# observation enabled on the processor. AERR# observation on the processor is enabled by asserting A8# on the active-to-inactive transition of RESET#.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G45. Processor May Assert DRDY# on a Write with No Data

Problem: When a MASKMOVQ instruction is misaligned across a chunk boundary in a way that one chunk has a mask of all 0's, the processor will initiate two partial write transactions with one having all byte enables deasserted. Under these conditions, the expected behavior of the processor would be to perform both write transactions, but to deassert DRDY# during the transaction which has no byte enables asserted. As a result of this erratum, DRDY# is asserted even though no data is being transferred.

Implication: The implications of this erratum depend on the bus agents ability to handle this erroneous DRDY# assertion. If a bus agent cannot handle a DRDY# assertion in this situation, or attempts to use the invalid data on the bus during this transaction, unpredictable system behavior could result.

Workaround: A system which can accept a DRDY# assertion during a write with no data will not be affected by this erratum. In addition, this erratum will not occur if the MASKMOVQ is aligned.

Status: For the steppings affected see the *Summary Table of Changes* at the beginning of this section.

G46. Thermal Sensor Leakage Current May Exceed Specification

Problem: The thermal sensor in the Pentium III Xeon processor cartridge violates the maximum input leakage current specification in Table 9 of the *Pentium® III Xeon™ Processor at 500 and 550 MHz* datasheet (10 µA).

Implication: The thermal sensor incorporates input protection diodes on the SMBCLK and SMBDAT signals for ESD protection. The protection diodes can potentially clamp these lines to ~0.6 V when the $V_{CC_{SMBus}}$ voltage supply to the cartridge is powered off. Hence, when $V_{CC_{SMBus}}$ is powered down, the Pentium III Xeon processor thermal sensor may prevent SMBus transactions originating from SMBus devices powered by a different voltage source, but on the same SMBus, from occurring. SMBus devices that are powered from different voltage power planes are not typically located on the same SMBus. Designs using this isolation technique will not be affected by this erratum.

Workaround: If it is desired to have SMBus devices powered by a source other than $V_{CC_{SMBus}}$, these devices must be isolated from the Pentium III Xeon processor SMBus to ensure that this erratum does not occur.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G47. GP# Fault on WRMSR to ROB_CR_BKUPTMPDR6

Problem: Writing a '1' to unimplemented bit(s) in the ROB_CR_BKUPTMPDR6 MSR (offset 1E0h) will result in a general protection fault (GP#).

Implication: The normal process used to write an MSR is to read the MSR using RDMSR, modify the bit(s) of interest, and then to write the MSR using WRMSR. Because of this erratum, this process may result in a GP# fault when used to modify the ROB_CR_BKUPTMPDR6 MSR.

Workaround: When writing to ROB_CR_BKUPTMPDR6 all unimplemented bits must be '0.' Implemented bits may be set as '0' or '1' as desired.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G48. Heavy L2 Cache Traffic Results in Noise on the TCK Signal

Problem: Pentium III Xeon processors under heavy L2 cache traffic may induce noise on the TCK signal. If the system is also accessing the processor through the Test Access Port, erroneous operation may occur.

Implication: Access to the Test Access Port may fail if access is simultaneous with heavy L2 cache traffic.

Workaround: Ensure that any access to the Test Access Port does not occur simultaneously with heavy L2 cache traffic.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G49. Machine Check Exception May Occur Due to Improper Line Eviction In The IFU

Problem: The Pentium III Xeon processor is designed to signal an unrecoverable Machine Check Exception (MCE) as a consistency checking mechanism. Under a complex set of circumstances involving multiple speculative branches and memory accesses, there exists a one-cycle long window in which the processor may signal a MCE in the Instruction Fetch Unit (IFU) because instructions previously decoded have been evicted from the IFU. The one cycle long window is opened when an opportunistic fetch receives a partial hit on a previously executed but not as yet completed store resident in the store buffer. The resulting partial hit erroneously causes the eviction of a line from the IFU at a time when the processor is expecting the line to still be present. If the MCE for this particular IFU event is disabled, execution will continue normally.

Implication: While this erratum may occur on a system with any number of Pentium III Xeon processors, the probability of occurrence increases with the number of processors. If this erratum does occur, a machine check exception will result. Note systems that implement an operating system that does not enable the Machine Check Architecture will be completely unaffected by this erratum (e.g., Windows* 95 and Windows 98).

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G50. Machine Check Exception May Occur During L2 Cache Initialization

Problem: It is possible for a multiple bit ECC error to be induced by the random state of the TAG bits in the processor's cache component(s) at power up. If this multiple bit ECC error occurs, it may, depending on system configuration, result in a machine check exception error occurring during cache initialization.

Implication: This erratum may occur on a system with Pentium III Xeon processors which utilize TAG ECC within the cache component(s). The occurrence of this erratum is dependent on the specific cache initialization sequence. If this erratum occurs, a machine check exception will result.

Workaround: Executing the following code sequence in BIOS prior to enabling the processor's L1 cache will ensure that this erratum does not occur. In the case of a multiprocessor system, ensure that the BIOS workaround code executes on each processor individually prior to enabling the L1 cache:

```
                PUSHAD                                  ; Save registers.
        ;-------------------------------------------------------------------
        ; Get feature bits now and check for processor type (family/model).
        ;-------------------------------------------------------------------
                MOV     EAX, 1h                         ; CPUID version/feature parameter.
                CPUID                                   ; Execute CPUID instruction.
                CMP     AX, 672h                        ; Check for Pentium III Xeon processor, Model 7,
                JL      SkipWorkaround                  ; If not B0 stepping or greater of this processor..
                CMP     AX, 680h                        ; ..exit, skipping the workaround.
                JGE     SkipWorkaround
        ;-------------------------------------------------------------------
        ; Check Platform ID bits now and only execute code if Slot 2.
        ;-------------------------------------------------------------------
                MOV     ECX, BBL_CR_OVRD    ; MSR 17h
                RDMSR
                AND     EDX, 001C0000h                  ; check Platform ID bits to ensure that ..
                CMP     EDX, 00080000h                  ; .. it is the SC330 version of the CPUID..
                JNE     SkipWorkaround                  ; .. If not, exit
        ;-------------------------------------------------------------------------------------------
        ; Ensure that L2 cache is disabled. The L2 cache is disabled by clearing BBL_CR_CTL3 bit 8.
        ; Also, we write BBL_CR_CTL3  controlled L2 features to default WITHOUT disturbing
        ; RESERVED or read only register bits. The previous BBL_CR_CTL3  settings are restored
        ; on completion.
        ;-------------------------------------------------------------------------------------------

                MOV     ECX, BBL_CR_CTL3    ; MSR 11Eh
                RDMSR
                PUSH    EAX                             ; Save original value.
                PUSH    EDX
                AND     EAX, 0FF881E00h                 ; Clear address space, disable CSER checking, clear cache
; size, disable L2, disable CRTN parity checking, disable
                                                        ; victim address parity checking, disable ECC, L2 not
                                                        ; configured, and clear latency setting
                OR      EAX, 00001001Fh                 ; Set 2MB, latency, and L2 configured
                WRMSR
                INVD                                    ; Flush L2 tags.
                POP     EDX                             ; Restore original value
                POP     EAX
                WRMSR
        SkipWorkaround:

POPAD                                   ; Restore registers.
```

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G51. Snoop Request May Cause DBSY# Hang

Problem: A small window of time exists in which a snoop request originating from a bus agent to a processor with one or more outstanding memory transactions may cause the processor to assert DBSY# without issuing a corresponding bus transaction, causing the processor to hang (livelock). The exact circumstances are complex, and include the relative timing of internal processor functions with the snoop request from a bus agent.

Implication: This erratum may occur on a system with any number of processors. However, the probability of occurrence increases with the number of processors. If this erratum does occur, the system will hang with DBSY# asserted. At this point, the system requires a hard reset.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G52. Performance Counters Include Streaming SIMD Extensions L1 Prefetch

Problem: The processor allows the measurement of the frequency and duration of numerous different internal and bus related events (see the *Intel Architecture Software Developer's Manual, Volume 3,* for more details). The Streaming SIMD Extension (SSE) architecture provides a mechanism to pre-load data into the L1 cache, bypassing the L2 cache. The number of these L1 pre-loads measured by the performance monitoring logic will incorrectly be included in the count of “L2_LINES_IN” (24H) events and “L2_LINES_OUT” (26H) events.

Implication: If application software is run which utilizes the SSE L1 prefetch feature, the count of “L2_LINES_IN” (24H) and “L2_LINES_OUT” (26H) will read a value that is greater than the correct value.

Workaround: The correct value of “L2_LINES_IN” and “L2_LINES_OUT” may be calculated by subtracting the value of the “MMX_PRE_MISS” (4BH) from each of these registers.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G53. Lower Bits of SMRAM SMBASE Register Cannot Be Written With an ITP

Problem: The System Management Base (SMBASE) register (7EF8H) stores the starting address of the System Management RAM (SMRAM). This register is used by the processor when it is in System Management Mode (SMM), and its contents serve as the memory base for code execution and data storage. The 32-bit SMBASE register can normally be programmed to any value. When programmed with an In-Target Probe (ITP), however, any attempt to set the lower 11 bits of SMBASE to anything other than zeros via the WRMSR instruction will cause the attempted write to fail.

Implication: When set via an ITP, any attempt to relocate SMRAM space must be made with 2-Kbyte alignment.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G54. Task Switch May Cause Wrong PTE and PDE Access Bit to be Set

Problem: If an operating system executes a task switch via a Task State Segment (TSS), and the TSS is wholly or partially located within a clean page (A and D bits clear) and the GDT entry for the new TSS is either misaligned across a cache line boundary or is in a clean page, the accessed and dirty bits for an incorrect page table/directory entry may be set.

Implication: An operating system which uses hardware task switching (or hardware task management) may encounter this erratum. The effect of the erratum depends on the alignment of the TSS and ranges from no anomalous behavior to unexpected errors.

Workaround: The operating system could align all TSSs to be within page boundaries and set the A and D bits for those pages to avoid this erratum. The operating system may alternately use software task management.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section**.**

G55. Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results

Problem: The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC.

Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected instruction execution from the processor that is executing the modified code.

Implication: In this case, the phrase "unexpected execution behavior" encompasses the generation of most of the exceptions listed in the *Intel Architecture Software Developer's Manual Volume 3: System Programming Guide* including a General Protection Fault (GPF). In the event of a GPF the application executing the unsynchronized XMC operation would be terminated by the operating system.

Workaround: In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the *Intel Architecture Software Developer's Manual Volume 3: System Programming Guide*, Section 7.1.3.

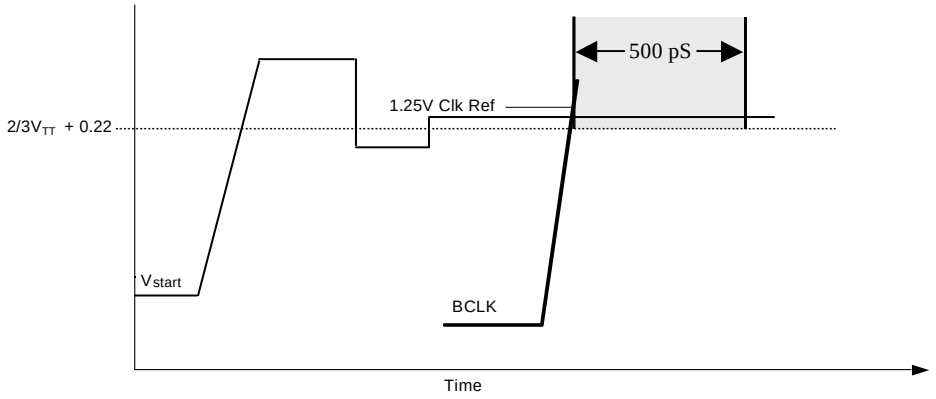
Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G56. V_{IH} Specification Deviation on Processor RS0# Input

Problem: The RS0# signal of the 550 MHz Pentium III Xeon processor with 512K and 1M of cache has been found to deviate from the published specification for the minimum input high voltage, $V_{IH(MIN)}$, as published in the *Pentium III Xeon™ Processor at 500 MHz and 550 MHz Datasheet.*

The published AGTL+ AC specification for $V_{IH(MIN)}$ is $(2/3 \cdot V_{TT}) + 0.1$ Volts.

Intel has characterized the new AGTL+ AC specification for $V_{IH(MIN)}$ on the processor's RS0# signal to be: $(2/3 \cdot V_{TT}) + 0.22$ Volts within 500 picoseconds following the rising edge of BCLK crossing the 1.25-Volt threshold. See below.



Implication: The deviation from the published specification results in decreased noise margin on the RS0# processor input. This decreased noise margin has been observed to be particularly detrimental in system environments where the RS0# processor input is being driven by an I/O buffer with a pull-up device (such as a p-channel transistor) that is switched off mid-cycle. Systems that employ I/O buffers that keep the pull-up device on for the entire cycle are significantly less affected by this erratum. The reduction in noise margin may lead to the generation of BINIT# (when enabled) resulting in a system reset, otherwise this condition may lead to a system "hang" condition.

Workaround: If a system design cannot meet the specification deviation, the following areas should be carefully examined to assure that additional margin exists in order to compensate for any deficit observed with respect to the newly characterized $V_{IH(MIN)}$ specification:

1. Consider increasing V_{TT} to a higher voltage than 1.5 Volts. Higher V_{TT} leads to improved noise margin by increasing the difference between V_{TT} and V_{REF}. For each 10-mV increase in V_{TT}, an additional 3.3-mV improvement in margin is achieved. Care should be exercised so as to avoid exceeding the maximum V_{TT} specification of 1.635 Volts.

2. For crow's foot front side bus topology, reduce the value of the termination resistor attached to the RS0# network at the main trunk from the original recommended value of 150Ω to 50Ω. If no resistor exists at this junction, a 50-Ω resistor may be added at this point. Other topologies should be examined to determine if reduction in the termination impedance at the source achieves additional margin to the newly characterized $V_{IH(MIN)}$ specification.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G57. Noise Sensitivity Issue on Processor SMI# Pin

Problem: Post silicon characterization has demonstrated a greater than expected sensitivity to noise on the processor's SMI# input, which may result in spurious SMI# interrupts.

Implication: BIOS/SMM code that is capable of handling spurious SMI events will report a spurious SMI#, but should not be negatively impacted by this erratum. Systems whose BIOS code cannot handle spurious SMI events may fail, resulting in a system hang or other anomalous behavior.
Spurious SMI# interrupts should be controlled on the system board regardless of BIOS implementation.

Workaround: Possible workarounds that may reduce or eliminate the occurrence of the spurious SMI include:

1. Use a lower effective pull-up resistance on the SMI# pin. This resistor must meet the specifications of the component driving the SMI# signal.
2. Externally condition the SMI# signal prior to providing it to the processor's SMI# pin.

These workarounds should be evaluated on a design-by-design basis.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G58. Processor Will Erroneously Report a BIST Failure

Problem: If the processor performs BIST at power-up, the EAX register is normally cleared (0H) when the processor passes. The processor will erroneously report a non-zero value (signaling a BIST failure) even if BIST passes.

Implication: The processor will incorrectly signal an error after BIST is performed.

Workaround: The system BIOS should ignore the BIST results in the EAX register.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G59. L2_LD and L2_M_LINES_OUTM Performance-Monitoring Counters Do Not Work

Problem: The L2_LD (29H) and L2_M_LINES_OUTM (27H) Performance-Monitoring counters are used to monitor L2 cache line activity. These counters incorrectly count their respective events.

Implication: These counters will report incorrect data.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G60. Limitation on Cache Line ECC Detection and Correction

Problem: ECC can detect and correct up to four single-bit ECC errors per cache line. However, the processor will only detect and correct one single-bit ECC error per cache line.

Implication: The processor may report fewer single-bit ECC errors and more double-bit ECC errors than previous processors.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G61. IFU/DCU Deadlock May Cause System Hang

Problem: An internal deadlock situation may occur in systems with multiple bus agents, with a failure signature such that a processor either asserts DBSY# without issuing the corresponding data, or fails to respond to a snoop request from another bus agent. Should this erratum occur, the affected processor ceases code execution and the system will hang.

The specific circumstances surrounding the occurrence of this erratum are:

1. A locked operation to the Data Cache Unit (DCU) is in process.
2. A snoop occurs, but cannot complete due to the ongoing locked operation.
3. The presence of the snoop prevents pending Instruction Fetch Unit (IFU) requests from completing.
4. The IFU requests are periodically restarted.

The continued IFU restart attempts create additional DCU snoops, which prevent the in-process locked operation from completing, keeping the DCU locked.

Implication: The system may hang.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G62. L2_DBUS_BUSY Performance Monitoring Counter will not Count Writes

Problem: The L2_DBUS_BUSY (22H) performance monitoring counter is intended to count the number of cycles during which the L2 data bus is in use. For some steppings of the processor, the L2_DBUS_BUSY counter will not be incremented during write cycles and therefore will only reflect the number of L2 data bus cycles resulting from cache reads.

Implication: The L2_DBUS_BUSY event counts only L2 read cycles.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G63. Deadlock May Occur Due to Illegal-Instruction/Page-Miss Combination

Problem: Intel's 32-bit Instruction Set Architecture (ISA) utilizes most of the available op-code space; however some byte combinations remain undefined and are considered illegal instructions. Intel processors detect the attempted execution of illegal instructions and signal an exception. This exception is handled by the operating system and/or application software.

Under a complex set of internal and external conditions involving illegal instructions, a deadlock may occur within the processor. The necessary conditions for the deadlock involve:

1. The illegal instruction is executed.
2. Two page table walks occur within a narrow timing window coincident with the illegal instruction.

Implication: The illegal instructions involved in this erratum are unusual and invalid byte combinations that are not useful to application software or operating systems. These combinations are not normally generated in the course of software programming, nor are such sequences known by Intel to be generated in commercially available software and tools. Development tools (compilers, assemblers) do not generate this type of code sequence, and will normally flag such a sequence as an error. If this erratum occurs, the processor deadlock condition will occur and result in a system hang. Code execution cannot continue without a system RESET.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G64. Incorrect Sign May Occur On X87 Result Due To Indefinite QNaN Result From Streaming SIMD Extensions Multiply

Problem: It is possible that a negative sign bit may be incorrectly applied to the result of an X87 floating-point operation if it is closely preceded by a Streaming SIMD Extensions (SSE) multiply operation. In order for this erratum to occur, the Streaming SIMD Extensions multiply operation must result in an Indefinite Quiet Not-a-Number (QNaN). Operations such as multiplying zero by infinity will result in an Indefinite QNaN result.

Implication: If this erratum occurs, the result of an X87 floating-point instruction which should be positive will instead be negative.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G65. MASKMOVQ Instruction Interaction with String Operation May Cause Livelock

Problem: Under the following scenario, combined with a specific alignment of internal events, the processor may enter a deadlock condition:

1. A store operation completes, leaving a write-combining (WC) buffer partially filled.
2. The target of a subsequent MASKMOVQ instruction is split across a cache line.
3. The data in (2) above results in a hit to the data in the WC buffer in (1).

Implication: If this erratum occurs, the processor deadlock condition will occur and result in a system hang. Code execution cannot continue without a system RESET.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G66. FLUSH# Assertion Following STPCLK# May Prevent CPU Clocks From Stopping

Problem: If FLUSH# is asserted after STPCLK# is asserted, the cache flush operation will not occur until after STPCLK# is de-asserted. Furthermore, the pending flush will prevent the processor from entering the Sleep state, since the flush operation must complete prior to the processor entering the Sleep state.

Implication: Following SLP# assertion, processor power dissipation may be higher than expected.

Workaround: For systems that use the FLUSH# input signal and Deep Sleep state of the processor, ensure that FLUSH# is not asserted while STPCLK# is asserted.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G67. Floating-Point Exception Condition May be Deferred

Problem: A floating-point instruction that causes a pending floating-point exception (ES=1) is normally signaled by the processor on the next waiting FP/MMX™ technology instruction. In the following set of circumstances, the exception may be delayed or the FSW register may contain a wrong value:

1. The excepting floating-point instruction is followed by an instruction that accesses memory across a page (4-Kbyte) boundary or its access results in the update of a page table dirty/access bit.
2. The memory accessing instruction is immediately followed by a waiting floating-point or MMX technology instruction.
3. The waiting floating-point or MMX technology instruction retires during a one-cycle window that coincides with a sequence of internal events related to instruction cache line eviction.

Implication: The floating-point exception will not be signaled until the next waiting floating-point/MMX technology instruction. Alternatively it may be signaled with the wrong TOS and condition code values. This erratum has not been observed in any commercial software applications.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G68. Intermittent Failure to Assert ADS# During System Power-On

Problem: Under a system specific set of initial parametric conditions, a very small number of Pentium® III Xeon™ processors (CPUID 068xh) can be susceptible to entering an internal test mode during processor power-on. The symptom of this test mode is a failure to assert ADS# during a processor power-on.

Implication: On susceptible platforms, when power is applied to the processor, there is a possibility that the processor will occasionally enter the test mode rather than initiate a system boot sequence.

Workaround: A subsequent processor Power-Off then Power-On cycle should remove the processor from this test mode, allowing normal processor operation to resume. The following workaround also can reduce the occurrence of the failure condition, and is already implemented on the SC330 cartridge:

SC330-based platform designs in which VTT leads the processor core voltage (SC330 pin B83) can reduce the occurrence of the erratum by connecting SC330 pin B34 (RESERVED) to pin B7 (VTT). Note that this connection is already implemented on the SC330 cartridge.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G69. Race Conditions May Exist On Thermal Sensor SMBus Collision Detection/Arbitration Circuitry

Problem: In certain SMBus configurations, when the thermal sensor is used in “hard wired alert” mode along with at least one other device on the bus, the thermal sensor may continue to send its address after losing a collision arbitration in response to an Alert Response Address (ARA) by the SMBus controller.

In order for this erratum to occur, all of the following conditions must be present:

1. The thermal sensor must be configured with alert enabled (default setting).
2. There must be one or more other devices on the SMBus along with the thermal sensor.
3. One or more of these other devices must be also configured with alert enabled.
4. One or more of these other devices must have a lower address (higher priority) than the thermal sensor.
5. The thermal sensor must generate an SM alert while at least one other device has an SM alert pending to be serviced.

In this situation, the thermal sensor will continue to send its address on the SMBus even if it has lower priority than the pending alert. When this occurs, the SMBus controller cannot correctly interpret the device address. This may cause the thermal sensor’s alert flag not to clear and may result in SMBus lockup.

Implication: The SMBus controller may see an invalid address and the resulting response of the SMBus controller will vary from implementation to implementation.

Workaround: Remove any of the five conditions listed above or:

1. In software, use polling mode for the thermal sensor data collection with alert disabled. This software workaround has been validated on both Intel's test platforms as well as on certain OEM systems.
2. Ensure that the thermal sensor alert may be cleared by a hardware or software mechanism. The implementation of this workaround will be system dependent.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G70. Cache Line Reads May Result in Eviction of Invalid Data

Problem: A small window of time exists in which internal timing conditions in the processor cache logic may result in the eviction of an L2 cache line marked in the invalid state.

Implication: There are three possible implications of this erratum:

1. The processor may provide incorrect L2 cache line data by evicting an invalid line.
2. A BNR# (Block Next Request) stall may occur on the system bus.
3. Should a snoop request occur to the same cache line in a small window of time, the processor may incorrectly assert HITM#. It is then possible for an infinite snoop stall to occur should another processor respond (correctly) to the snoop request with HIT#. In order for this infinite snoop stall to occur, at least three agents must be present, and the probability of occurrence increases with the number of processors.

Should 2 or 3 occur, the processor will eventually assert BINIT# (if enabled) with an MCA error code indicating a ROB time-out. At this point, the system requires a hard reset.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Changes* at the beginning of this section.

G71. Receiver Noise Sensitivity May Result in System Bus Single-Bit ECC Errors

Problem: System bus single-bit ECC errors may be detected and corrected in systems with excessive falling edge ringback noise or noise induced from multi-bit switching (ground bounce, crosstalk, etc…) on data signals already held in an electrically low state. In certain cases this falling edge ringback may trigger the AGTL+ receiver's dynamic clamp circuit, which may activate at a reduced input voltage level (approximately 100-200 mV lower than the expected 1.0V level).

Implication: If the amplitude of the noise on the victim line reaches approximately 0.9-1.0V, then the processor receiver may interpret this noise as a rising edge event and activate the dynamic clamp. If the dynamic clamp pulls the low signal's voltage up to the V_{IL} (input low voltage) level, then the receiver may interpret this electrically low signal as a high value (assuming this V_{IL} violation occurs within the setup time window). Should a single-bit ECC error occur, the correctable ECC error bit will be asserted in the MC*i*_STATUS register. In systems with inadequate noise-margin on AGTL+ signaling, it is possible that double-bit or multiple-bit ECC errors may occur. In this event, a system hang or "blue screen" would occur. The impact of this erratum is highly dependent on system bus design as it relates to the amount of noise inherent in the bus layout.

Workaround: The AGTL+ V_{REF} (input voltage reference) level will be raised from 1.0V to 1.1V (assuming a V_{TT} of 1.5V) on the processor cartridge substrate. This allows the dynamic clamp to activate at a trigger level approximately 100 mV higher than with the previous 1.0V V_{REF} setting. Due to the dependence upon system bus design, bus layout may be evaluated to determine the need for improvements in falling edge ringback levels. This may involve one or several of the following baseboard modifications:

1. Change termination resistor value to reduce multi-bit noise and/or improve V_{IL} noise margin.
2. Reduce multi-bit noise by improving ground return paths (additional high-frequency decoupling, improved AC plane referencing, etc.).
3. Raising or lowering the V_{TT} level (within the AC & DC tolerances specified in the processor EMTS) to improve V_{IL} noise margin and/or reduce the multi-bit noise due to dI/dt transients.

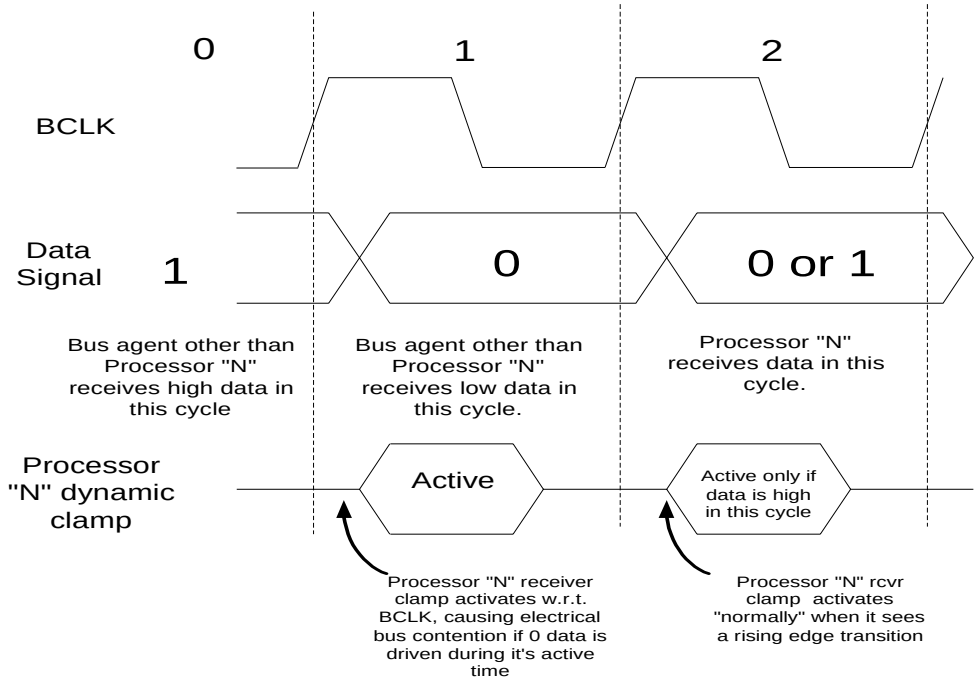
Stress tests using various data patterns and transactions should be run to identify the worst-case multi-bit noise conditions for this erratum.

Status: For the steppings affected, see the *Summary of Changes* at the beginning of this section.

G72. AGTL+ Receiver May Induce Falling Edge Ledges and Undershoot Levels

Problem: Falling edge ledge push-out and undershoot may be seen on AGTL+ data and data bus ECC signals. The AGTL+ receiver's dynamic clamp circuit activates (i.e. clamps the signal to an electrically high level) in certain cycles when data is driven to an electrically low level by another agent.

Implication: This erratum only occurs on a processor that does not receive data in that particular cycle. More specifically, this erratum only occurs when a specific processor does not receive data in the current BCLK period, but receives data in the very next period. Therefore, the processor inducing this falling edge noise (on data and data bus ECC signals) will not actually read the data in that cycle. See below for an illustration of this erratum affecting a processor denoted Processor "N". The falling edge ledges and undershoots may affect the signal quality and/or timing at agents that receive data in that cycle. Therefore, the impact of this erratum is highly dependent on system bus design.



Workaround: Evaluate system bus signal quality and timing margin at the valid receiving agent as a result of this falling edge noise. Due to dependence upon system bus design, bus layout may be evaluated to determine the need for improvements in signal quality and timing margin at the valid receiving agent as a result of this receiver-induced falling edge noise.

Status: For the steppings affected, see the *Summary of Changes* at the beginning of this section.

G73. Snoop Probe During FLUSH# Could Cause L2 to be Left in Shared State

Problem: During a L2 FLUSH operation using the FLUSH# pin, it is possible that a read request from a bus agent or other processor to a valid line will leave the line in the Shared state (S) instead of the Invalid state (I) as expected after flush operation. Before the FLUSH operation is completed, another snoop request to

invalidate the line from another agent or processor could be ignored, again leaving the line in the Shared state.

Implication: Current desktop and mid range server systems have no mechanism to assert the flush pin and hence are not affected by this errata. A high-end server system that does not suppress snoop traffic before the assertion of the FLUSH# pin may cause a line to be left in an incorrect cache state.

Workaround: Affected systems (those capable of asserting the FLUSH# pin) should prevent snoop activity on the front side bus until invalidation is completed after asserting FLUSH#, or use a WBINVD instruction instead of asserting the FLUSH# pin in order to flush the cache.

Status: For the steppings affected, see the *Summary of Changes* at the beginning of this section.

G74. Livelock May Occur Due to IFU Line Eviction

Problem: Following the conditions outlined for erratum G49, if the instruction that is currently being executed from the evicted line must be restarted by the IFU, and the IFU receives another partial hit on a previously executed (but not as yet completed) store that is resident in the store buffer, then a livelock may occur.

Implication: If this erratum occurs, the processor will hang in a live lock-situation, and the system will require a reset to continue normal operation.

Workaround: None identified

Status: For the steppings affected, see the *Summary of Changes* at the beginning of this section.

G75. L2 Cache May Not Be Correctly Initialized Following a Power-On Reset

Problem: Following a power-on reset, it is possible that an uninitialized internal node in the processor may prevent the processor's L2 cache from correctly initializing.

Implication: With random data in the L2 TAG RAM and Cache RAM, two behaviors may occur:

1. Execution of a WBINVD will cause any line whose TAG incorrectly indicates an M-state to be written to memory, potentially causing a system hang; and
2. Memory accesses that incorrectly result in a cache 'HITM#' may receive a line of random data from the cache, and memory accesses that result in either a cache 'HIT#' or 'HITM#' may result in the logging and signaling of an uncorrectable ECC error.

Workaround: There are two identified workarounds:

1. The first workaround, involving placing the processor into hardware BIST by ensuring that the INIT# signal is asserted on the trailing edge of RESET# at power-on, will ensure that the cache is correctly initialized and will completely resolve this issue.
2. The second workaround greatly reduces, but does not completely eliminate, the exposure to this issue. It involves:

a) Execution of an INVD instruction on the bootstrap processor (BSP) before processor caching is enabled, and as early as possible within BIOS.

b) Initiation of a second processor reset by the BSP as closely as possible to the INVD instruction in step 2a above.

The INVD instruction causes all L2 cache MESI status flags to be set to the I-state, reducing the chance for false cache "Hits". The second reset allows the cache to be correctly initialized, which limits the exposure of this issue to the period of time between the power-on reset and the second reset. The BIOS can typically assert the RESET# signal by writing to a chipset configuration register.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G76. Selector for the LTR/LLDT Register May Get Corrupted

Problem: The internal selector portion of the respective register (TR, LDTR) may get corrupted if, during a small window of LTR or LLDT system instruction execution, the following sequence of events occur:

1. Speculative write to a segment register that might follow the LTR or LLDT instruction
2. The read segment descriptor of LTR/LLDT operation spans a page (4 Kbytes) boundary; or causes a page fault

Implication: Incorrect selector for LTR, LLDT instruction could be used after a task switch.

Workaround: Software can insert a serializing instruction between the LTR or LLDT instruction and the segment register write.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G77. INIT Does Not Clear Global Entries in the TLB

Problem: INIT may not flush a TLB entry when:

1. The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register);
2. G bit for the page table entry is set;
3. TLB entry is present in TLB when INIT occurs.

Implication: Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

Workaround: Write to CR3, CR4 or CR0 registers before writing to memory early in BIOS code to clear all the global entries from TLB.

Status: For the steppings affected see the Summary of Changes at the beginning of this section.

G78. VM Bit Will Be Cleared on a Double Fault Handler

Problem: Following a task switch to a Double Fault Handler that was initiated while the processor was in virtual-8086 (VM86) mode, the VM bit will be incorrectly cleared in EFLAGS.

Implication: When the OS recovers from the double fault handler, the processor will no longer be in VM86 mode.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G79. Memory Aliasing With Inconsistent A and D Bits May Cause Processor Deadlock

Problem: In the event that software implements memory aliasing by having two Page Directory Entries (PDEs) point to a common Page Table Entry (PTE) and the Accessed and Dirty bits for the two PDEs are allowed to become inconsistent the processor may become deadlocked.

Implication: This erratum has not been observed with commercially available software.

Workaround: Software that needs to implement memory aliasing in this way should manage the consistency of the Accessed and Dirty bits.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G80. Use of Memory Aliasing With Inconsistent Memory Type May Cause System Hang

Problem: Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang. This would occur if one of the addresses is non-cacheable used in code segment and the other a cacheable address. If the cacheable address finds its way in instruction cache, and non-cacheable address is fetched in IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will expect this instruction to still be in fetch unit and lack of it will cause system hang.

Implication: This erratum has not been observed with commercially available software.

Workaround: Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged this practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G81. Processor May Report Invalid TSS Fault Instead of Double Fault During Mode-C Paging

Problem: When an operating system executes a task switch via a Task State Segment (TSS) the CR3 register is always updated from the new task TSS. In the mode C paging, once the CR3 is changed the processor will attempt to load the PDPTRs. If the CR3 from the target task TSS or task switch handler TSS is not valid then the new PDPTR will not be loaded. This will lead to the reporting of invalid TSS fault instead of the expected double fault.

Implication: Operating systems that access an invalid TSS may get invalid TSS fault instead of a double fault.

Workaround: Software needs to ensure any accessed TSS is valid.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G82. Machine Check Exception May Occur When Interleaving Code Between Different Memory Types

Problem: A small window of opportunity exists where code fetches interleaved between different memory types may cause a machine check exception. A complex set of micro-architectural boundary conditions is required to expose this window.

Implication: Interleaved instruction fetches between different memory types may result in a machine check exception. The system may hang if machine check exceptions are disabled. Intel has not observed the occurrence of this erratum while running commercially available applications or operating systems.

Workaround: Software can avoid this erratum by placing a serializing instruction between code fetches which span different memory types.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G83. Wrong ESP Register Values During a Fault in VM86 Mode

Problem: At the beginning of the IRET instruction execution in VM86 mode, the lower 16 bits of the ESP register are saved as the old stack value. When a fault occurs, these 16 bits are moved into the 32-bit ESP, effectively clearing the upper 16 bits of the ESP.

Implication: This erratum has not been observed to cause any problems with commercially available software.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G84. APIC ICR Write May Cause Interrupt Not to be Sent When ICR Delivery Bit Pending

Problem: If the APIC ICR (Interrupt Control Register) is written with a new interrupt command while the Delivery Status bit from a previous interrupt command is set to '1' (Send Pending), the interrupt message may not be sent out by the processor.

Implication: This erratum will cause an interrupt message not to be sent, potentially resulting in system hang.

Workaround: Software should always poll the Delivery Status bit in the APIC ICR and ensure that it is '0' (Idle) before writing a new value to the ICR.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G85. High Temperature and Low Supply Voltage Operation May Result in Incorrect Processor Operation

Problem: When operating at the high temperature, low supply voltage corner of the processor specification, if there is a store pending in the processor's fill buffer, and simultaneously a load operation misses the L1 cache but results in a hit to the L2 cache, then it is possible that incorrect data may be returned to satisfy the load operation.

Implication: When this erratum is encountered, unpredictable software behavior may occur. It can be seen from the table of affected steppings that this erratum is constrained to a single stepping and is only possible in processors operating at frequencies of 933MHz and above and is not present in all of those processors. Application of the workaround will prevent occurrence of the erratum in all processors of that stepping.

Workaround: It is possible for BIOS code to contain a workaround for this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G86. The Instruction Fetch Unit (IFU) May Fetch Instructions Based Upon Stale CR3 Data After a Write to CR3 Register

Problem: Under a complex set of conditions, there exists a one-clock window following a write to the CR3 register wherein it is possible for the iTLB fill buffer to obtain a stale page translation based on the stale CR3 data. This stale translation will persist until the next write to the CR3 register, the next page fault or execution of a certain class of instructions including CPUID or IRETD with privilege level change.

Implication: The wrong page translation could be used leading to erroneous software behavior.

Workaround: Operating systems that are potentially affected can add a second write to the CR3 register.

Status: For the stepping affected see the Summary of Changes at the beginning of this section.

G87. Under Some Complex Conditions, the Instructions in the Shadow of a JMP FAR may be Unintentionally Executed and Retired

Problem: If all of the following events happen in sequence it is possible for the system or application to hang or to execute with incorrect data.
1. The execution of an instruction, with an OPCODE that requires the processor to stall the issue of micro-instructions in the flow from the microcode sequence logic block to the instruction decode block (a StallMS condition).
2. Less than 63 (39 for Pre-CPUID 0x6BX) micro-instructions later, the execution of a mispredictable branch instruction (Jcc, LOOPcc, RET Near, CALL Near Indirect, JMP ECX=0, or JMP Near Indirect).
3. The conditional branch in event (2) is mispredicted, and furthermore the mispredicted path of execution must result in either an ITLB miss, or an Instruction Cache miss. This needs to briefly stall the issue of micro-instructions again immediately after the conditional branch until that branch prediction is corrected by the jump execution block (a 2nd StallMS condition).

4. Along the correct path of execution, the next instruction must contain a 3rd StallMS condition at a precisely aligned point in the execution of the instruction (CLTS, POPSS, LSS, or MOV to SS).
5.A JMP FAR instruction must execute within the next 63 micro-instructions (39 Pre-CPUID 0x6BX). The intervening micro-instructions must not have any events or faults.
When the instruction from event (2) retires, the StallMS condition within the event (5) instruction fails to operate correctly, and instructions in the shadow of the JMP FAR instruction could be unintentionally executed.

Implication: Occurrence of this erratum could lead to erroneous software behavior. Intel has not identified any commercially available software which may encounter this condition; this erratum was discovered in a focused test environment. One of the four instructions that are required to trigger this erratum, CLTS, is a privileged instruction that is only executed by an operating system or driver code. The remaining three instructions, POPSS, LSS, and MOV to SS, are executed infrequently in modern 32-bit application code.

Workaround: None identified at this time.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G88. Exx. Processor Does not Flag #GP on Non-zero Write to Certain MSRs

Problem: When a non-zero write occurs to the upper 32 bits of SYSENTER_EIP_MSR or SYSENTER_ESP_MSR, the processor should indicate a general protection fault by flagging #GP. Due to this erratum, the processor does not flag #GP.

Implication: The processor unexpectedly does not flag #GP on a non-zero write to the upper 32 bits of SYSENTER_EIP_MSR or SYSENTER_ESP_MSR. No known commercially available operating system has been identified to be affected by this erratum.

Workaround: None identified.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G89 POPF and POPFD Instructions that Set the Trap Flag Bit May Cause Unpredictable Processor Behavior

Problem: In some rare cases, POPF and POPFD instructions that set the Trap Flag (TF) bit in the EFLAGS register (causing the processor to enter Single-Step mode) may cause unpredictable processor behavior.

Implication: Single step operation is typically enabled during software debug activities, not during normal system operation.

Workaround: There is no workaround for single step operation in commercially available software. For debug activities on custom software, the POPF and POPFD instructions could be immediately followed by a NOP instruction to facilitate correct execution.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G90 FXSAVE after FNINIT Without an Intervening FP (Floating Point) Instruction May Save Uninitialized Values for FDP (x87 FPU Instruction Operand (Data) Pointer Offset) and FDS (x87 FPU Instruction Operand (Data) Pointer Selector)

Problem: An FXSAVE after FNINIT without an intervening FP instruction may save uninitialized values for FDP and FDS.

Implication: When this erratum occurs, the values for FDP/FDS in the FXSAVE structure may appear to be random values. These values will be initialized by the first FP instruction executed after the FXRSTOR that restor the saved floating point state. Any FP instruction with memory operand will initialize FDP/FDS. Intel has not observed this erratum with any commercially available software.

Workaround: After an FINIT, do not expect the FXSAVE memory image to be correct, until at least one FP instruction with a memory operand has been executed.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G91 FSTP (Floating Point Store) Instruction Under Certain Conditions May Result in Erroneously Setting a Valid Bit on an FP (Floating Point) Stack Register

Problem: An FSTP instruction with a PDE/PTE (Page Directory Entry/Page Table Entry) A/D bit update followed by user mode access fault due to a code fetch to a page that has supervisor-only access permission may result in erroneously setting a valid bit of an FP stack register. The FP top of stack pointer is unchanged.

Implication: This erratum may cause an unexpected stack overflow.

Workaround: User mode code should not count on being able to recover from illegal accesses to memory regions protected with supervisor-only access when using FP instructions.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G92 Page with PAT (Page Attribute Table) Set to USWC (Uncacheable Speculative Write Combine) While Associated MTRR (Memory Type Range Register) is UC (Uncacheable) May Consolidate to UC

Problem: A page whose PAT memory type is USWC while the relevent MTRR memory type is UC, the consolidated memory type may be treated as UC (rather than WC as specified in the IA-32 Intel® Architecture Software Developer's Manual).

Implication: When this erratum occurs, the memory page may be marked as UC (rather than WC). This may have a negative performance impact.

Workaround: None identified.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G93 Under Certain Conditions LTR (Load Task Register) Instruction May Result in System Hang

Problem: An LTR instruction may result in a system hang if all the following conditions are met:

1. Invalid data selector of the TR (Task Register) resulting in either #GP (General Protection Fault) or #NP (Segment Not Present Fault).
2. GDT (Global Descriptor Table) is not 8-byte aligned.
3. Data BP (breakpoint) is set on cache line containing the descriptor data.

Implication: This erratum may result in system hang if all conditions have been met. This erratum has not been observed in commercial operating systems or software. For performance reasons, GDT is typically aligned to 8 bytes.

Workaround: Software should align GDT to 8 bytes.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G94 Load from Memory Type USWC (Uncacheable Speculative Write Combine) May Get Its Data Internally Forwarded from a Previous Pending Store

Problem: A load from memory type USWC may get its data internally forwarded from a pending store. As a result, the expected load may never be issued to the external bus.

Implication: When this erratum occurs, a USWC load request may be satisfied without being observed on the external bus. There are no known usage models where this behavior results in any negative side-effects.

Workaround: Do not use memory type USWC for memory that has read side-effects.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G95 Code Segment Limit Violation May Occur on 4 Gigabyte Limit Check

Problem: Code Segment limit violations may occur on 4 Gigabyte limit check when the code stream wraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: Avoid code that wraps around segment limit.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G96 FST Instruction with Numeric and Null Segment Exceptions May Cause General Protection Faults to be Missed and FP Linear Address (FLA) Mismatch

Problem: An FST instruction combined with numeric and null segment exceptions may cause General Protection Faults to be missed and FP Linear Address (FLA) mismatch.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: None identified.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G97 Code Segment (CS) is Incorrect on SMM Handler when SMBASE is not Aligned

Problem: With SMBASE being relocated to a non-aligned addres, during SMM entry the CS can be improperly updated which can lead to an incorrect SMM handler.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: Align SMBASE to 32 Kbytes.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

G1AP. APIC Access to Cacheable Memory Causes Shutdown

Problem: APIC operations which access memory with any type other than uncacheable (UC) are illegal. If an APIC operation to a memory type other than UC occurs and Machine Check Exceptions (MCEs) are disabled, the processor will enter shutdown after such an access. If MCEs are enabled, an MCE will occur. However, in this circumstance, a second MCE will be signaled. The second MCE signal will cause the Pentium III Xeon processor to enter shutdown.

Implication: Recovery from a PIC access to cacheable memory will not be successful. Software that accesses only UC type memory during APIC operations will not encounter this erratum.

Workaround: Ensure that the memory space to which PIC accesses can be made is marked as type UC (uncacheable) in the memory type range registers (MTRRs) to avoid this erratum.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G2AP. MP Systems May Hang Due to Catastrophic Errors During BSP Determination

Problem: In MP systems, a catastrophic error during the bootstrap processor (BSP) determination process should cause the assertion of IERR#. If the catastrophic error is due to the APIC data bus being stuck at electrical zero, then the system hangs without asserting IERR#.

Implication: MP systems may hang during boot due to a catastrophic error. This erratum has not been observed to date in a typical commercial system, but was found during focused system testing using a grounded APIC data bus.

Workaround: None identified

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G3AP. Write to Mask LVT (Programmed as EXTINT) Will Not Deassert Outstanding Interrupt

Problem: If the APIC subsystem is configured in Virtual Wire Mode implemented through the local APIC, (i.e., the 8259 INTR signal is connected to LINT0 and LVT1's interrupt delivery mode field is programmed as EXTINT), a write to LVT1 intended to mask interrupts will not deassert the internal interrupt source if the external LINT0 signal is already asserted. The interrupt will be erroneously posted to the Pentium III Xeon processor despite the attempt to mask it via the LVT.

Implication: Because of the masking attempt, interrupts may be generated when the system software expects no interrupts to be posted.

Workaround: Software can issue a write to the 8259A interrupt mask register to deassert the LINT0 interrupt level, followed by a read to the controller to ensure that the LINT0 signal has been deasserted. Once this is ensured, software may then issue the write to mask LVT entry 1.

Status: For the steppings affected see the *Summary of Changes* at the beginning of this section.

G4AP REP MOVS Operation in Fast string Mode Continues in that Mode When Crossing into a Page with a Different Memory Type

Problem: A fast "REP MOVS" operation will continue to be handled in fast mode when the string operation crosses a page boundary into an Uncacheable (UC) memory type. Also if the fast string operation crosses a page boundary into a WC memory region, the processor will not self snoop the WC memory region. This may eventually result in incorrect data for some of the bytes of the last cache line stored, if the cache line was previously cached as WB (through aliasing) and modified.

Implication: String elements should be handled by the processor at the native operand size in UC memory. In the event that the WB to WC aliasing case occurs, the end result could vary from normal software execution to potential software failure. Intel has not observed either aspects of this erratum in commercially available software.

Workaround: Software operating within Intel's recommendation will not require WB and WC memory aliased to the same physical address.

Status: For the steppings affected, see the Summary of Table of Changes.

G5AP The FXSAVE, STOS, or MOVS Instruction May Cause a Store Ordering Violation When Data Crosses a Page with a UC Memory Type

Problem: If the data from an FXSAVE, STOS or MOVS instruction crosses a page boundary from WB to UC memory type and this instruction is immediately followed by a second instruction that also issues a store to memory, the final data stores from both instructions may occur in the wrong order.

Implication: The impact of this store ordering behavior may vary from normal software execution to potential software failure. Intel has not observed this erratum in commercially available software.

Workaround: FXSAVE, STOS, or MOVS data must not cross page boundary from WB to UC memory type.

Status: For the steppings affected, see the Summary of Changes at the beginning of this section.

DOCUMENTATION CHANGES

The Documentation Changes listed in this section apply to the following documents:

- *P6 Family of Processors Hardware Developer's Manual*
- *Pentium® III Xeon™ Processor at 500 and 550 MHz* datasheet
- *Pentium® III Xeon™ Processor at 600 MHz to 1 GHz with 256KB L2 Cache* datasheet
- *Pentium® III Xeon™ Processor at 700 MHz with 1MB and 2MB L2 Cache* datasheet
- *Intel Architecture Software Developer's Manual, Volumes 1, 2, and 3*
- *P6 Family of Processors Hardware Developer's Manual*

All Documentation Changes will be incorporated into a future version of the appropriate Pentium III Xeon processor documentation.

SPECIFICATION CLARIFICATIONS

The Specification Changes listed in this section apply to the following documents:

- *P6 Family of Processors Hardware Developer's Manual*
- *Pentium® III Xeon™ Processor at 500 and 550 MHz* datasheet
- *Pentium® III Xeon™ Processor at 600 MHz to 1 GHz with 256KB L2 Cache* datasheet
- *Pentium® III Xeon™ Processor at 700 MHz with 1MB and 2MB L2 Cache* datasheet
- *Intel Architecture Software Developer's Manual, Volumes 1, 2, and 3*
- *P6 Family of Processors Hardware Developer's Manual*

All Specification Changes will be incorporated into a future version of the appropriate Pentium III Xeon processor documentation.

G1. Specification Clarification with respect to time stamp counter

In the "Debugging and Performance Monitoring" chapter (Section 15.8, Section 15.10.9 and Section 15.10.9.3) of the *IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide*, the Time Stamp Counter definition has been updated to include support for the future processors. This change will be incorporated in the next revision of the *IA-32 Intel® Architecture Software Developer's Manual*.

15.8 TIME-STAMP COUNTER

The IA-32 architecture (beginning with the Pentium processor) defines a time-stamp counter mechanism that can be used to monitor and identify the relative time occurrence of processor events. The counter's architecture includes the following components:

- **TSC flag —** A feature bit that indicates the availability of the time-stamp counter. The counter is available in an IA-32 processor implementation if the function CPUID.1:EDX.TSC[bit 4] = 1.

- **IA32_TIME_STAMP_COUNTER MSR** (called TSC MSR in P6 family and Pentium processors) **—** The MSR used as the counter.

- **RDTSC instruction —** An instruction used to read the time-stamp counter.

- **TSD flag —** A control register flag is used to enable or disable the time-stamp counter (enabled if CR4.TSD[bit 2] = 1).

The time-stamp counter (as implemented in the P6 family, Pentium, Pentium M, Pentium 4, and Intel Xeon processors) is a 64-bit counter that is set to 0 following a RESET of the processor. Following a RESET, the counter will increment even when the processor is halted by the HLT instruction or the external STPCLK# pin. Note that the assertion of the external DPSLP# pin may cause the time-stamp counter to stop.

Members of the processor families increment the time-stamp counter differently:

- For Pentium M processors (family [06H], models [09H, 0DH]); for Pentium 4 processors, Intel Xeon processors (family [0FH], models [00H, 01H, or 02H]); and for P6 family processors: the time-stamp counter increments with every internal processor clock cycle. The internal processor clock cycle is determined by the current core-clock to bus-clock ratio. Intel® SpeedStep® technology transitions may also impact the processor clock.

- For Pentium 4 processors, Intel Xeon processors (family [0FH], models [03H and higher]): the time-stamp counter increments at a constant rate. That rate may be set by the maximum core-clock to bus-clock ratio of the processor or may be set by the frequency at which the processor is booted. The specific processor configuration determines the behavior. Constant TSC behavior ensures that the duration of each clock tick is uniform and supports the use of the TSC as a wall clock timer even if the processor core changes frequency. This is the architectural behavior moving forward.

NOTE

To determine average processor clock frequency, Intel recommends the use of Performance Monitoring logic to count processor core clocks over the period of time for which the average is required. See Section 15.10.9 and Appendix A in this manual for more information.

The RDTSC instruction reads the time-stamp counter and is guaranteed to return a monotonically increasing unique value whenever executed, except for a 64-bit counter wraparound. Intel guarantees that the time-stamp counter will not wraparound within 10 years after being reset. The period for counter wrap is longer for Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Normally, the RDTSC instruction can be executed by programs and procedures running at any privilege level and in virtual-8086 mode. The TSD flag allows use of this instruction to be restricted to programs and procedures running at privilege level 0. A secure operating system would set the TSD flag during system initialization to disable user access to the time-stamp counter. An operating system that disables user access to the time-stamp counter should emulate the instruction through a user-accessible programming interface.

The RDTSC instruction is not serializing or ordered with other instructions. It does not necessarily wait until all previous instructions have been executed before reading the counter. Similarly, subsequent instructions may begin execution before the RDTSC instruction operation is performed.

The RDMSR and WRMSR instructions read and write the time-stamp counter, treating the time-stamp counter as an ordinary MSR (address 10H). In the Pentium 4, Intel Xeon, and P6 family processors, all 64-bits of the time-stamp counter are read using RDMSR (just as with RDTSC). When WRMSR is used to write the time-stamp counter on processors before family [0FH], models [03H, 04H]: only the low order 32-bits of the time-stamp counter can be written (the high-order 32 bits are cleared to 0). For family [0FH], models [03H, 04H]: all 64 bits are writeable.

15.10.9 COUNTING CLOCKS

The count of cycles, also known as clockticks, forms a the basis for measuring how long a program takes to execute. Clockticks are also used as part of efficiency ratios like cycles per instruction (CPI). Processor clocks may stop ticking under circumstances like the following:

- The processor is halted when there is nothing for the CPU to do. For example, the processor may halt to save power while the computer is servicing an I/O request. When Hyper-Threading Technology is enabled, both logical processors must be halted for performance-monitoring counters to be powered down.
- The processor is asleep as a result of being halted or because of a power-management scheme. There are different levels of sleep. In the some deep sleep levels, the time-stamp counter stops counting.

There are three ways to count processor clock cycles to monitor performance. These are:

- **Non-halted clockticks —** Measures clock cycles in which the specified logical processor is not halted and is not in any power-saving state. When Hyper-Threading Technology is enabled, this these ticks can be measured on a per-logical-processor basis.
- **Non-sleep clockticks —** Measures clock cycles in which the specified physical processor is not in a sleep mode or in a power-saving state. These ticks can**n**ot be measured on a logical-processor basis.
- **Time-stamp counter** — Some processor models permit clock cycles to be measured when the physical processor is not in deep sleep (by using the time-stamp counter and the RDTSC instruction). Note that such ticks cannot be measured on a per-logical-processor basis. See Section 10.8 for detail on processor capabilities.

The first two methods use performance counters and can be set up to cause an interrupt upon overflow (for sampling). They may also be useful where it is easier for a tool to read a performance counter than to use a time stamp counter (the timestamp counter is accessed using the RDTSC instruction).

For applications with a significant amount of I/O, there are two ratios of interest:

- **Non-halted CPI —** Non-halted clockticks/instructions retired measures the CPI for phases where the CPU was being used. This ratio can be measured on a logical-processor basis when Hyper-Threading Technology is enabled.
- **Nominal CPI —** Time-stamp counter ticks/instructions retired measures the CPI over the duration of a program, including those periods when the machine halts while waiting for I/O.

15.10.9.3 Incrementing the Time-Stamp Counter

The time-stamp counter increments when the clock signal on the system bus is active and when the sleep pin is not asserted. The counter value can be read with the RDTSC instruction.
The time-stamp counter and the non-sleep clockticks count may not agree in all cases and for all processors. See Section 10.8 for more information on counter operation.

SPECIFICATION CHANGES

The Specification Changes listed in this section apply to the following documents:

- *P6 Family of Processors Hardware Developer's Manual*
- *Pentium® III Xeon™ Processor at 500 and 550 MHz* datasheet
- *Pentium® III Xeon™ Processor at 600 MHz to 1 GHz with 256KB L2 Cache* datasheet
- *Pentium® III Xeon™ Processor at 700 MHz with 1MB and 2MB L2 Cache* datasheet
- *Intel Architecture Software Developer's Manual, Volumes 1, 2, and 3*
- *P6 Family of Processors Hardware Developer's Manual*

All Specification Changes will be incorporated into a future version of the appropriate Pentium III Xeon processor documentation.